

STPGA: Selection of training populations with a genetic algorithm

Deniz Akdemir

Michigan State University

Correspondence: deniz.akdemir.work@gmail.com

Abstract

Optimal subset selection is an important task that has numerous algorithms designed for it and has many application areas. **STPGA** contains a special genetic algorithm supplemented with a tabu memory property (that keeps track of previously tried solutions and their fitness for a number of iterations), and with a regression of the fitness of the solutions on their coding that is used to form the ideal estimated solution (look ahead property) to search for solutions of generic optimal subset selection problems. I have initially developed the programs for the specific problem of selecting training populations for genomic prediction or association problems, therefore I give discussion of the theory behind optimal design of experiments to explain the default optimization criteria in **STPGA**, and illustrate the use of the programs in this endeavor. Nevertheless, I have picked a few other areas of application: supervised and unsupervised variable selection based on kernel alignment, supervised variable selection with design criteria, influential observation identification for regression, solving mixed integer quadratic optimization problems, balancing gains and inbreeding in a breeding population. Some of these illustrations pertain new statistical approaches.

Keywords: Genomic Prediction, Genome-Wide Association, Optimal Design of Experiments, High Dimensional Data, Sparse Models, Markers, Anchor Markers, Inbreeding, Genetic Gain, Kinship, Similarity, Subset Selection, Mixed Integer Quadratic Programming, Genomic Selection, Supervised Unsupervised Variable Selection, R, Genomic Selection.

1. Introduction

The paper introduces the R (R Core Team 2016) package **STPGA** that provides a genetic algorithm for subset selection. The package is available from the Comprehensive R Archive Network (CRAN) at <http://CRAN.R-project.org/package=STPGA>, and some of the underlying motivations, methodology and results were presented in (Akdemir *et al.* 2015; Isidro *et al.* 2015; Crossa *et al.* 2016; Akdemir and Sánchez 2016) and also some innovations that

will be detailed in several subsequent articles. This document details version 4.0 which includes major upgrades and bug fixes compared to previous versions.

Numerous other algorithms have been proposed for the optimal subset selection problem, many of them are heuristic exchange type algorithms (Fedorov 1972; Mitchell 1974; Nguyen and Miller 1992; Rincent *et al.* 2012; Isidro *et al.* 2015). In exchange type algorithms new solutions are obtained by adding one point and removing another at a time (some exchange algorithms might allow exchange of more than one design point at once), these algorithms are greedy and are only proven to find the best subset for certain type of design criteria. In general, exchange algorithms are prone to getting stuck in local optimal solutions. Branch and bound (BB) (Furnival and Wilson 1974) is a global exhaustive search method that has proven to be reasonably efficient on practical problems. BB searches the design region by iteratively dividing design region and searching each piece for an optimal solution. BB is often more efficient than straight enumeration because it can eliminate regions that provably do not contain an optimal solution. Welch (1982) uses a BB algorithm to find globally best D -optimal design for a given design criteria and a set of candidate points. Another method that has been applied to the subset selection problem is simulated annealing (Haines 1987). Branch and bound and simulated annealing algorithms require appreciable computation time even for moderate sized problems.

Genetic algorithms (GAs) are a class of evolutionary algorithms made popular by John Holland and his colleagues (Goldberg and Holland 1988; Holland 1992a,b), and have been applied to find exact or approximate solutions to optimization and search problems (Goldberg and Holland 1988; Sivanandam and Deepa 2007; Akdemir *et al.* 2015; Akdemir and Sánchez 2016). There are numerous packages that implement GAs or similar evolutionary algorithms. The packages **gafit** (Tendys 2002), **galts** (Satman 2013), **genalg** (Willighagen 2005), **rgenoud** (Mebane Jr *et al.* 2015), **DEoptim** (Ardia *et al.* 2016) and the **GA** (Scrucca *et al.* 2013) offer many options for using optimization routines based on evolutionary algorithms. The optimization algorithm that is used in **STPGA** (LA-GA-T algorithm) is a modified genetic algorithm with tabu search and look ahead property and it is specialized for solving subset selection problems.

Today's trends in computation are towards computer architectures that integrate many, less complex processors, exploit thread-level and data-level parallelism. This makes these computers perfect ground for implementation of evolutionary algorithms for solving complex optimization problems since these algorithms can be easily to be run at parallel. To make my point more clear, lets remember Amdahl's law (Amdahl 1967) which puts a limit to the speed that can be gained by parallelizing a process:

$$\text{speedup} = \frac{\text{serial processing time}}{\text{parallel processing time}} \leq \frac{1}{f_{par}/N_p + (1 - f_{par})},$$

where f_{par} is the fraction of code which could be parallelized, $1 - f_{par}$ is the serial fraction and N_p is the number of processors. As it can be observed from the figures in Figure 1, obtained by applying this formula for varying values of f_{par} between 0 and 1 and values of N_p in $\{1, 2, 4, 8, 16, 32, 64, 128, 256\}$, under the assumption that the number of processors doubles each year and the processing capabilities for each parallel node and everything else identical throughout, taking full advantage of parallelization requires methods that have parallelization frequency close to one. From the same figure we can read that 1% change in parallelization frequency might cause up to 3.5 times speedup if there are 256 processors and a ideal parallelizable procedure might have speed more than to 50 times relative to a 80% parallelizable procedure. Evolutionary algorithms, like GA, fall at the very right end of these figures.

At the Michigan State University, at the beginning of year 2017, there were hundreds of nodes available for the researches through their high performance computer cluster (HPCC) system. It is not science fiction to claim that clusters with millions of nodes will be available to researchers with the technologies such as cloud / grid computing. We are faced with the challenge of matching these these parallel resources with methods that can use them efficiently.

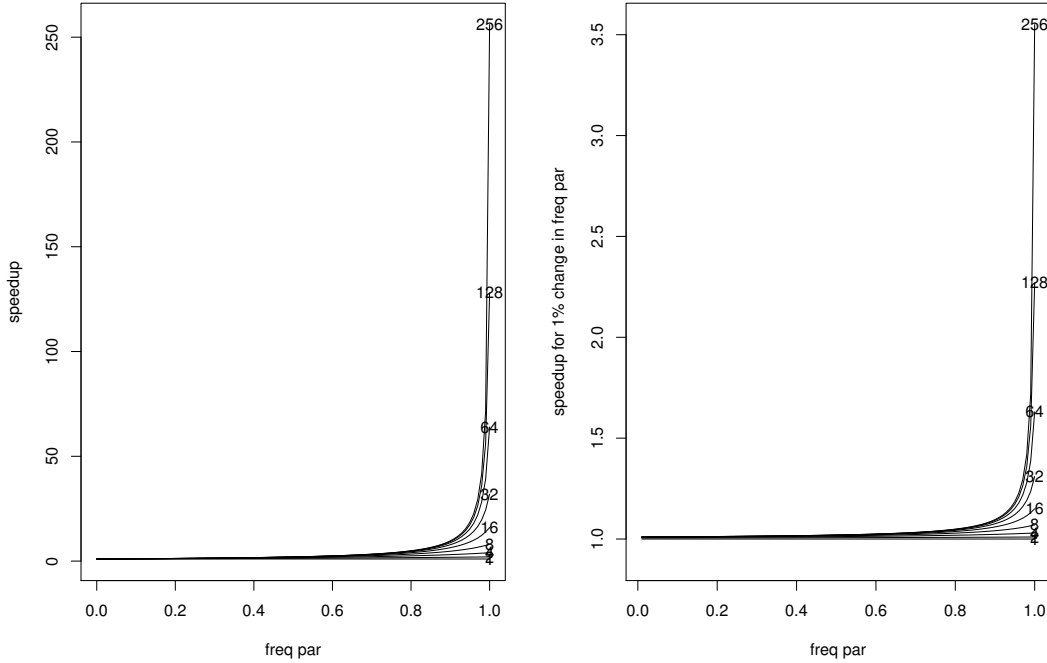


Figure 1: (Left) Speedup for $\{1, 2, 4, 8, 16, 32, 64, 128, 256\}$ processors for changing values of parallelization frequency, (Right) speedup for 1% change in parallelization frequency.

In my view, easy adaptation of GAs to parallel computation is a major advantage of GAs to other subset selection algorithms. GAs scale well with large and/or computationally expensive

problems and can achieve reasonable running times by using parallel computing architectures with either shared or distributed memory systems, these systems are becoming increasingly available to the researchers. Scientific community prefer algorithms that run faster in serial. However, the direction the improvements in the computer technology seem be more in the parallelization rather than faster processors.

Some advantages of the GAs to other subset selection algorithms include the following:

- GA can be applied to many different problems and does not need to be reinvented for each new problem.
- GA is a very flexible optimization algorithm, evolutionary mechanisms that are involved in GA can be modified at different stages of the algorithm; various selection strategies, various penalization of the objective function, can be explored simultaneously or in a serial fashion.
- Adopting GAs in a parallel computing environment is easy. This might involve, for example, evaluation of the fitness function for the current GA population, or running many GAs at parallel to provide initial solutions generation of GAs.

In addition the LA-GA-T algorithm in **STPGA** adds two more properties:

- Inferior solutions that were visited recently will not be visited. This property is akin to a memory in an intelligent system.
- The state of current solutions are used to predict the best ideal solution. This gives the algorithm the look ahead property. This property is akin to inference in an intelligent system.

Nature solves problems through evolutionary processes, it works with communities of solutions that exploits the their communal information content to create new solutions, it is this information that persists, not the individual solutions. In addition, we have long standing theories explaining how and why such evolutionary processes work. There is also a vast amount of principled study of evolutionary mechanisms, both that are natural and artificial; the whole subject of evolutionary genetics; the methodology, theories and practices related to breeding; the theoretical and practical approaches of evolutionary algorithms and computation allows us humans to understand and manage these systems.

In the next section, I briefly review the basic ideas behind simple GAs and the LA-GA-T that is used in STPGA. Then, I present the details of the interface to the **STPGA** package in Section 3, followed by several examples section and the conclusions. The examples section has been divided into two main parts: STPGA for selection of training populations, and STPGA in other subset selection problems. Both of these sections are rather long and detailed, especially

the part that relates to optimal design that introduces some concepts and ideas of optimal design of experiments with a focus on predictive learning using regression models. Some of the design criteria discussed in this section are implemented in **STPGA**, a table listing of these criteria is provided. I also demonstrate how to write user defined criteria.

2. Optimizer in STPGA

The optimization algorithm that is used in **STPGA** is a modified genetic algorithm with tabu search and look ahead property. Genetic algorithms are stochastic search algorithms which are able to solve optimization problems using evolutionary strategies inspired by the basic principles of biological evolution. They use a population of candidate solutions that are represented as binary strings of 0's and 1's, this population evolving toward better solutions. At each iteration of the algorithm, a fitness function is used to evaluate and select the elite individuals and subsequently the next population is formed from the elites by genetically motivated operations such as crossover and mutation. The properties and prospects of genetic algorithms were first laid out in the cornerstone book of Holland ([Holland 1992a](#)).

GAs have an implicit parallelism property ([Holland 1992a](#)). In addition, since GA uses a set of solutions at each iteration it couples well with the advanced computers (workstations with many processors and large memory) and computer systems (high performance computing clusters, cloud computing technologies) of today allowing it to be applied to very large scale optimization problems. In my opinion, with the advent of new technologies like DNA computing ([Liu et al. 2000](#); [Paun et al. 2005](#)) that uses programmable molecular computing machines or quantum computers ([Gruska 1999](#); [Leuenberger and Loss 2001](#)) that operate on "qubits", parallelizable algorithms such as GA will have more and more important role in big scale optimization problems. The GA algorithm in STPGA is supplemented with two additional principles, tabu (memory) and inference through prediction based on a current population of solutions. I refer to it as the LA-GA-T (look ahead genetic algorithm with tabu) algorithm.

Tabu search is a search where most recently visited solutions are avoided by keeping a track of the previously tried solutions. This avoids many function evaluations and decreases the number of iterations till convergence, it is especially useful for generating new solutions around local optima.

The LA-GA-T algorithm in **STPGA** also uses the binary coding of the current population of solutions and their fitness to fit a linear ridge regression model from which the effects of individual digits in this binary code are estimated assuming that the contribution of an individual to the criterion value does not change much in relation to different subsets. The predicted ideal solution based on this model is constructed and included in the elite population of solutions. This gives the algorithm a look ahead property and improves the speed of

convergence especially in the initial steps of the optimization. I should note here that the idea of regressing the fitnesses of solutions on their designs was inspired by the genomic selection methodology recently put into use in plant and animal breeding with the promise of increasing genetics gains from selection per unit of time.

As can be seen from the Figures 2 and 3, LA-GA-T converges in much fewer iterations compared to a simple GA. However, I have to note that the per iteration computation time for LA-GA-T algorithm is slightly higher compared to a simple GA.

Procedure 1 Genetic Algorithm

- 1: $t = 0$.
 - 2: initialization -Create an initial population of solutions of desired size, S_t .
 - 3: Memory for tabu is empty, $MemTabu_t = NULL$;
 - 4: **repeat**
 - 5: $t = t + 1$,
 - 6: Evaluation -For each solution in S_{t-1} calculate the criterion value,
 - 7: Look ahead -Use the binary coding of S_{t-1} and their fitness to fit a linear ridge regression model from which the effects of individual digits in this binary code are estimated. Put this solution in S_t ,
 - 8: Selection -Identify the best solutions by the ordering of criterion values, these are denoted by E_t ,
 - 9: Elitism -Let the best solution in E_t be s_t . Put s_t in S_t ,
 - 10: Tabu -Update memory for tabu by letting $MemTabu_t = S_{t-1}$.
 - 11: **repeat**
 - 12: Crossover-Randomly pick two solutions in E_t . Recombination of these two solutions are obtained by summing the frequency distributions of these solutions and sampling with new solutions using probabilities corresponding to this combined frequency distribution.
 - 13: Mutation - With a given probability decrease the frequency of a mate that has positive frequency by some integer value less than the current frequency of that mate and increase the frequency of some other mate pair is by the same amount.
 - 14: **if** the resulting solution is in $MemTabu_t$ **then**
 - 15: eliminate solution
 - 16: **else**
 - 17: Insert solution into S_t .
 - 18: **until** S_t has N_{pop} solutions.
 - 19: **until** Convergence is reached
 - 20: Evaluation -For each solution in S_t calculate the criterion value,
 - 21: Selection - Identify the best solutions by the ordering of criterion values, these are denoted by E_t ,
 - 22: Elitism -Let the best solution in E_t be s_t . Return s_t .
-

The solutions obtained by any run of GA may be sub-optimal and different solutions can be obtained given different starting populations. Another layer of safety is obtained if the algorithm is started from multiple initial populations and an island model of evolution is used where separate populations are evolved independently for several steps and then the best

solutions from these algorithms becomes the initial solutions to evolutionary algorithm. Since the functions in STPGA can start from user provided initial values, island models and other strategies can be combined when using the algorithm. I give an example code for doing this in the Appendix section.

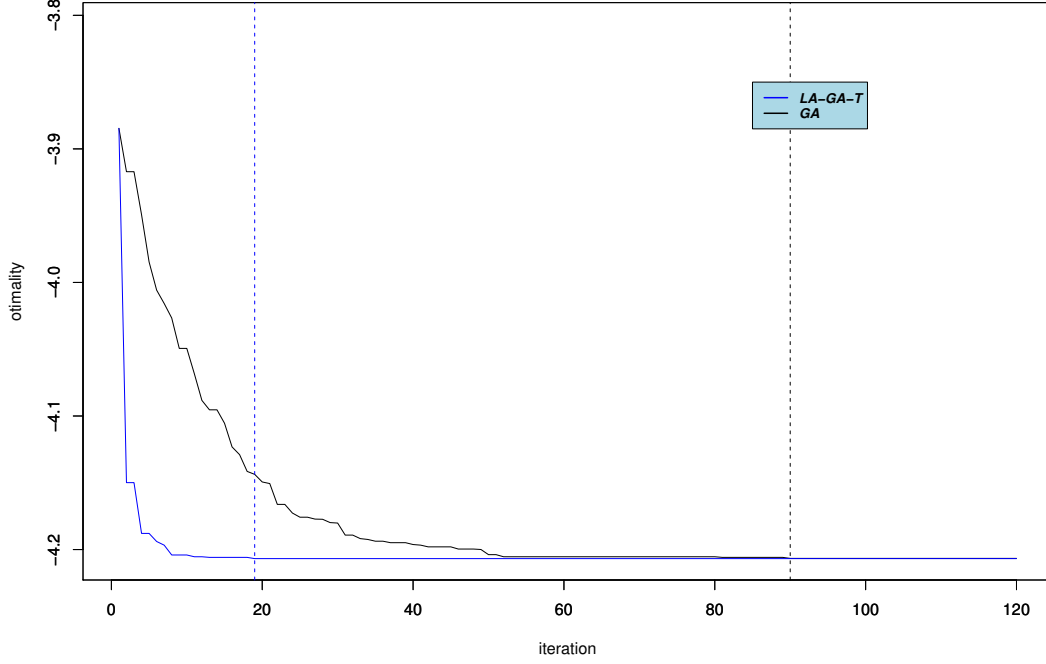


Figure 2: The convergence of GA and LA-GA-T for no test scenario.

3. Software interface, computational considerations

There are two main functions in **STPGA**, these are **GenAlgForSubsetSelection** and **GenAlgForSubsetSelectionNoTest**. The function **GenAlgForSubsetSelection** uses a simple genetic algorithm to identify a training set of a specified size from a larger set of candidates which minimizes an optimization criterion (for a known test set). The function **GenAlgForSubsetSelectionNoTest** is for identifying a training set of a specified size from a larger set of candidates which minimizes an optimization criterion, no test set is specified. These functions share a lot of common parameters, except **GenAlgForSubsetSelection** requires an additional input that specifies the target set of individuals and the data matrix should be supplemented to include the observed value of the variables for these target individuals. The inputs for these functions are described below:

Inputs

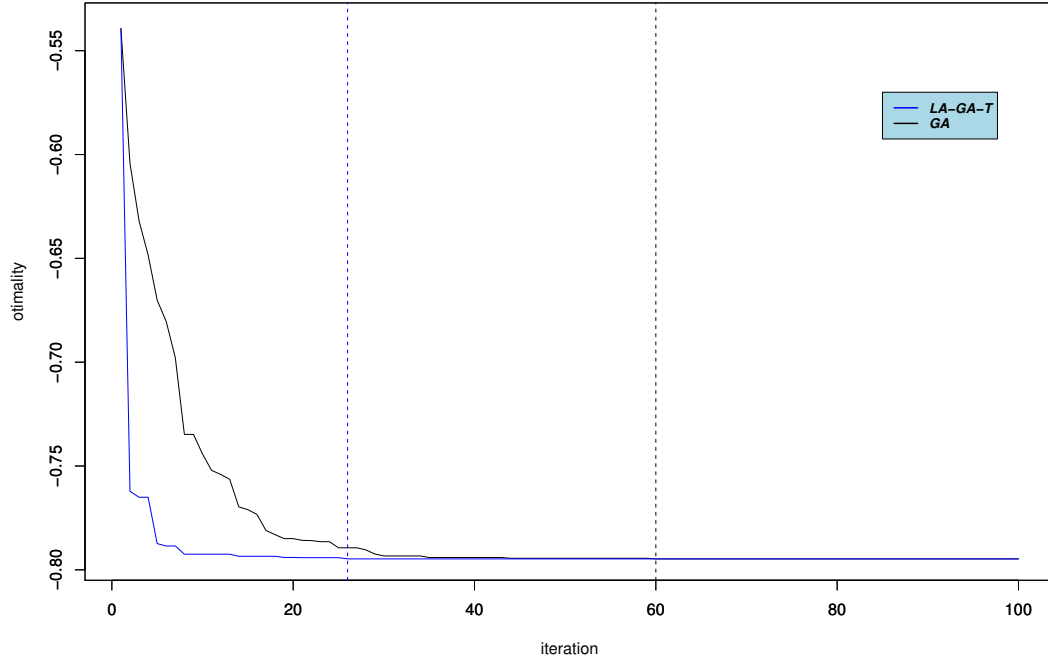


Figure 3: The convergence of GA and LA-GA-T when there is a target.

- **P** depending on the criterion this is either a numeric data matrix or a symmetric similarity matrix. When it is a data matrix, the union of the identifiers of the candidate (and test) individuals should be put as row names (and column names in case of a similarity matrix). For methods using the relationships, this is the inverse of the relationship matrix with row and column names as the the identifiers of the candidate (and test) individuals.
- **Candidates** vector of identifiers for the individuals in the candidate set.
- **Test** vector of identifiers for the individuals in the test set.
- **ntoselect** n_{Train} : number of individuals to select in the training set.
- **npop** genetic algorithm parameter, number of solutions at each iteration
- **nelite** genetic algorithm parameter, number of solutions selected as elite parents which will generate the next set of solutions.
- **keepbest** genetic algorithm parameter, TRUE or FALSE. If TRUE then the best solution is always kept in the next generation of solutions (elitism).

- **tabu** genetic algorithm parameter, TRUE or FALSE. If TRUE then the solutions that are saved in tabu memory will not be retried.
- **tabumemsize** genetic algorithm parameter, integer>0. Number of generations to hold in tabu memory.
- **mutprob** genetic algorithm parameter, probability of mutation for each generated solution.
- **mutintensity** mean of the Poisson variable that is used to decide the number of mutations for each cross.
- **niterations** genetic algorithm parameter, number of iterations.
- **minibefstop** genetic algorithm parameter, number of iterations before stopping if no change is observed in criterion value.
- **niterreg** genetic algorithm parameter, number of iterations to use regressions
- **lambda** scalar shrinkage parameter ($\lambda > 0$).
- **plotiters** plot the convergence: TRUE or FALSE. Default is FALSE.
- **plottype** type of plot, default is 1. possible values 1,2,3.
- **errorstat** optimality criterion: One of the optimality criterion. Default is "PEVMEAN". It is possible to use user defined functions as shown in the examples.
- **mc.cores** number of cores to use.
- **InitPop** a list of initial solutions
- **tolconv** if the algorithm cannot improve the errorstat more than tolconv for the last minibefstop iterations it will stop.
- **C** contrast matrix.
- **Vg** covariance matrix between traits generated by the relationship K (only for multi-trait version of PEVMEANMM).
- **Ve** residual covariance matrix for the traits (only for multi-trait version of PEVMEANMM).

All these inputs except **P**, **ntoselect** (also **Candidates** and **Test** for **GenAlgForSubsetSelection**) have default values of NULL meaning that they are internally assigned to the default suggested settings. These settings are as follows: npop = 100, nelite = 5, keepbest = TRUE, tabu = FALSE, tabumemsize = 1, mutprob = .8, mutintensity = 1, niterations = 500, minibefstop=100, niterreg = 5, lambda = 1e-6, plotiters = FALSE, plottype = 1, errorstat =

"PEVMEAN", $C = \text{NULL}$, $\text{mc.cores} = 1$, $\text{InitPop} = \text{NULL}$, $\text{tolconv} = 1\text{e-}7$, $\text{Vg} = \text{NULL}$, $\text{Ve} = \text{NULL}$. In a specific application of **STPGA**, we recommend the users to change these options until they are satisfied with the final results. Especially, when used with large data sets (many columns or rows), the parameters npop , niterations , minitbefstop should be increased. Both functions return a named list of length $\text{nelite} + 1$. The first nelite elements of the list are optimized training samples of size n_{train} and they are listed in increasing order of the optimization criterion. The last item on the list is a vector that stores the minimum values of the objective function at each iteration. The solution with best criterion value has name 'Solution with rank 1', the second 'Solution with rank 2', etc, ... The minimum values of the objective function through the iterations has name 'Best criterion values over iterations'.

The function **GenAlgForSubsetSelection** in the package uses this algorithm to identify a training set of a specified size from a larger set of candidates which minimizes an optimization criterion (for a known test set). The function "GenAlgForSubsetSelectionNoTest" tries to identify a training set of a specified size from a larger set of candidates which minimizes an optimization criterion, no test set is specified.

The subset selection algorithms in "GenAlgForSubsetSelectionNoTest" and "GenAlgForSubsetSelection" have somewhat different inner workings. "GenAlgForSubsetSelectionNoTest" splits the individuals given in row names of the input matrix \mathbf{P} into two parts: a set called Train of size "ntoselect" and its complement. The "GenAlgForSubsetSelection" starts with an input defining of split of the individuals given in row names of the input matrix \mathbf{P} into three parts: a set called "Candidates", a set called "Test" and their complement, after this the algorithm splits the set "Candidates" into a set called "Train" and its complement.

These two functions can be used with any user defined fitness functions and in the examples section, I will illustrate how these mechanisms can be used for general subset selection problems.

I have developed this package for my interest in solving certain design problems. Therefore, I have included several of my favorite design criteria in **STPGA**. A list of the names, required input parameters and the corresponding formulas are summarized with the Table 1 for reference. More explanation about the usage, examples and other details of these criterion can be found in the package help documentations.

When using a design criterion that uses the design matrix of the target individuals along with the candidates, the "GenAlgForSubsetSelection" function uses the individuals listed in "Test" extract the design matrix of these individuals from the design matrix of all individuals \mathbf{P} . If you use a similar criterion with the "GenAlgForSubsetSelection" function 'the design of the target set is implicitly assigned as the rows in \mathbf{P} not in "Train".

Many modern statistical learning problems involve the analysis of high dimensional data. For example, in genomic prediction problems phenotypes are regressed on large numbers of

genome-wide markers. **STPGA** was initially prepared for working with high dimensional data related to such whole-genome-regression (Meuwissen *et al.* 2001; Akdemir and Jannink 2015; Daetwyler *et al.* 2013) and association (Risch *et al.* 1996; Burton *et al.* 2007; Rietveld *et al.* 2013; Tian *et al.* 2011) approaches that are becoming increasingly popular for the analysis and prediction of complex traits in plants (e.g. Crossa *et al.* 2010), animals (e.g. VanRaden *et al.* 2009; Hayes *et al.* 2009) and humans (e.g. Yang *et al.* 2010; Makowsky *et al.* 2011). The design criteria in **STPGA** and their use were motivated by the practical problem of selecting the best genotypes for a phenotypic experiment so that the inferences made based on the data obtained by the experiment are optimally informative for genomic prediction and association problems (Isidro *et al.* 2015; Akdemir *et al.* 2015; Crossa *et al.* 2016). These high dimensional design problems pose additional computational challenges and the selection and use of the design criteria has a big influence on computational requirements. I recommended the use of dimension reduction techniques, whether they are supervised, unsupervised or based on algebraic manipulation, such as the use of dimension reduction methods like principle components analysis or variable clustering, use of methods based on similarity or distance matrices, before running **STPGA**. Several design optimization criteria in **STPGA** are equivalent and will produce the same or similar designs. However, calculation of one might be easier than the other based on the relative number of rows or columns of the data matrix.

STPGA software is written purely in R, however the computationally demanding criteria can be programmed by the user in C, C++ or Fortran. **STPGA** also benefits from multi-thread computing. The computational performance of the algorithm can be greatly improved if R is linked against a tuned BLAS implementation with multi-thread support, for example OpenBLAS, ATLAS, Intel mkl, etc.

4. Illustrations

In this section, I am going to illustrate use of **STPGA**. The first group of examples are related to selection of training populations. The second part, includes examples of other subset selection problems.

4.1. STPGA for selection of training populations

Experiments provide useful information to scientists as long as they are properly designed and analysed. The history of the theoretical work on the design problem goes way back. Fisher (Fisher 1992, 1960) gives a mathematical treatment of the determination of designs for some models. The first extended presentation of the ideas of optimum experimental designs appear in (Kiefer 1959) and (Yates 1935). A brief history of statistical work on optimum experimental design is given by Wynn (wynn1984jack) and the subject continues to develop, recently at an increasing rate. Box et al. (Box *et al.* 1978), Box and Draper (Draper and

Pukelsheim 1996; Atkinson and Donev 1992; Pukelsheim 2006) are a few of the authoritative texts on the subject. The alphabetical naming of designs is due to (Kiefer *et al.* 1985). For a detailed discussion of standard criteria reference is made to these references.

In this paper we focus only on a the narrow design problem of selecting an optimal set of n design points, $X_{Train} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, from a set of candidate design points $X_C = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$. The design defined by these n points, can be viewed as a measure on the candidate set $X_C = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$. Let ζ be a probability measure on X_C such that $\zeta(x_i) = 0$ if $x_i \notin X_{Train}$, and $\zeta(x_i) = 1/n$ if $x_i \in X_{Train}$.

When dealing with problems of supervised learning where the resulting model of the experiment will be used to make inferences about a known set of individuals, we can distinguish between the candidate set and a target set $X_{Target} = \{\mathbf{x}_1, \dots, \mathbf{x}_t\}$: X_{Target} describes the focused design region for which predictions about the dependent variables based on the models trained on X_{Train} are required. Let's assume that $1 \leq n \leq N$ and $1 \leq t$ are fixed integers, x_i are p -vectors and we denote the matrix form of X_{Train} as X , this is called the design matrix; and X_{Target} as X^* , this is called the design matrix for the target space.

The first component of a design optimization problem is the objective function. For example the objective function might be chosen as theoretically or numerically obtained sampling variance of a prespecified estimator of a population quantity of interest. The second component of an optimization problem is the set of decision variables and the constraints on the values of these variables. Once the objective function and the set of constraints are known the next step is to use a method to look for solutions that optimize the objective function and also adhering to the constraints.

Parametric design criteria usually depend on a function of the information matrix for the model parameters that gives some indication about the sampling variance and covariance of the estimated parameters. Let $I_{\theta}(\zeta)$ denote the information matrix of the parameters θ for a given design ζ . In order to be able to achieve a criteria that orders designs with respect to their information matrices, usually, a scalar function of the information matrix is used. These designs criteria have alphabetical names, the designs obtained by optimizing these criteria are referred to as A-, D, E-, G-, etc,... optimal designs. The list of design criteria that are implemented in **STPGA** are described by Table 1 with references to the equations in this manuscript from which these were inspired.

Many practical and theoretical problems in science treat relationships of the type $y = g(\mathbf{x}, \theta)$, where the response, y , is thought of as a particular value of a real-valued model function or response function, g , evaluated at the pair of arguments (\mathbf{x}, θ) . The parameter value θ , unknown to the experimenter, is assumed to lie in a parameter domain Θ . This is called the regression of y on \mathbf{x} . The **STPGA** is not confined to regression, but we use regression analysis to do most of the explaining and demonstrations.

Linear models

The choice of the function g is central to the regression model building process. One of the simplest regression models is the linear model. Let $X_{n \times p}$ be the design matrix, $\beta_{p \times 1}$ the vector of regression parameters, $y_{n \times 1}$ the vector of observations, and $\epsilon_{n \times 1} = (\epsilon_1, \epsilon_2, \dots, \epsilon_n)'$ our error vector giving

$$y = X\beta + \epsilon.$$

With I_n as the $n \times n$ identity matrix, the model is represented by the expectation vector and covariance matrix of y ,

$$E(y) = X\beta, \text{cov}(y) = \sigma^2 I_n,$$

and is termed the classical linear model with moment assumptions. We assume ϵ_i , $i = 1, 2, \dots, n$ will be iid with mean zero and $\text{cov}(\epsilon) = \sigma^2 I_n$. Under the additional normality assumption we write $y \sim N(0, \sigma^2 I_n)$.

We now concentrate on determining the optimal estimator for $c'\beta$ in the linear regression model. If X is not of full rank, it is not possible to estimate β uniquely. However, $X\beta$ is uniquely estimable, and so is $c'X\beta$ for any conformable vector c that is in the row space of X . If estimability holds then the Gauss-Markov Theorem determines the optimal estimator for $c'\beta$ to be $c'(X'X)^-X'y$, where A^- denotes any generalized inverse of A that satisfies $A = AA^-A$. The variance of this estimator depends only on the matrix $X'X$,

$$\text{var}_{\beta, \sigma^2}(c'(X'X)^-X'Y) = (\sigma^2)c'(X'X)^-X'X(X'X)^-c.$$

Up to the common factor σ^2/n , the optimal estimator has variance $c'(X'X)^-X'X(X'X)^-c$. Assuming estimability, the optimal estimator for the linear function of the coefficients $\gamma = C\beta$ is also given by the Gauss-Markov Theorem: $\hat{\gamma} = C(X'X)^-X'Y$. The covariance matrix of the estimator $\hat{\gamma}$ is $C(X'X)^-X'X(X'X)^-C'$. The covariance matrix becomes invertible provided the coefficient matrix C has full row rank.

A closely related task is that of prediction. Suppose we wish to predict additional responses $y^* = X^*\beta + \epsilon^*$. If we take the random vector $\hat{\gamma}$ from above as a predictor for the random vector y^* , to obtain precise estimators for $X^*\beta$, we would like to choose the design so as to maximize the relevant information matrix (minimize the covariance matrix). For example, G -optimal designs are obtained by minimizing the maximum variance of the predicted values, i.e., the maximum entry in the diagonal of the matrix $X(X'X)^-X'$.

Ridge Regression

Note the following: If A is a symmetric matrix, then the limit

$$\lim_{\lambda \rightarrow 0} (A + \lambda^2 I)^{-1}$$

is a generalized inverse of A , and also

$$\lim_{\lambda \rightarrow 0} (A + \lambda^2 I)^{-1} A \lim_{\lambda \rightarrow 0} (A + \lambda^2 I)^{-1} = \lim_{\lambda \rightarrow 0} (A + \lambda^2 I)^{-1}.$$

This means, for small $\lambda > 0$, $\hat{\gamma} \approx C(X'X + \lambda I)^{-1} X'Y$.

This estimator is called the ridge estimator, the coefficients have covariance matrix approximately proportional to

$$C(X'X + \lambda I)^{-1} C'. \quad (1)$$

Furthermore, prediction error variance for estimating the $CX^*\beta$ with ridge regression is approximately proportional to

$$CX^*(X'X + \lambda I)^{-1} X'^* C'. \quad (2)$$

Ridge estimators have smaller variance than BLUE's but they are biased since the estimators are "shrunk" towards zero.

Ridge estimators are especially useful when $X'X$ is singular. In some cases, the ridge estimation is only applied to a subset of the explanatory variables in X , for example it is customary to not shrink the mean term.

Splitting the design matrix X as $X = (X_F, X_R)$, where X_F contains the effects modeled without ridge penalty and X_R contains the terms modeled with ridge penalty, a design criterion concerning the estimation of shrunk coefficients can be written as

$$C(X'M^{-1}X + \lambda^2 I)^{-1} C'$$

with $M = I - X_F(X_F'X_F)^{-}X_F'$.

RKHS

Using the matrix identity $(P^{-1} + B'R^{-1}B)^{-1}B'R^{-1} = PB'(BPB' + R)^{-1}$, we can write $(X'X + \lambda I)^{-1}X' = X'(XX' + \lambda I)^{-1}$. The ridge regression solution for γ can then be written as follows:

$$\hat{\gamma} = C\hat{\beta} \approx C(X'X + \lambda I)^{-1}X'Y = CX'(XX' + \lambda I)^{-1}Y.$$

The important message here is that we only need access partitions of the matrix

$$K_{C,X}(\zeta) = \begin{bmatrix} XX' & XC' \\ CX' & CC' \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} \\ K_{12}' & K_{22} \end{bmatrix}$$

since $\hat{\gamma} = CX'(XX' + \lambda I)^{-1}Y = K_{21}(K_{11} + \lambda I)^{-1}Y$. Using

$$(X'X + \lambda I)^{-1} = \frac{1}{\lambda}(\mathbf{I} - X'(XX' + \lambda I)^{-1}X$$

we have

$$\begin{aligned}
C(XX + \lambda \mathbf{I})^{-1}C' &= C(\lambda(\frac{X'X}{\lambda} + \mathbf{I}))^{-1}C' \\
&= \frac{1}{\lambda}C(\mathbf{I} - X'(XX' + \lambda \mathbf{I})^{-1}X)C' \\
&= \frac{1}{\lambda}[CC'] - \frac{1}{\lambda}[CX'(XX' + \lambda \mathbf{I})^{-1}XC'] \\
&\propto \mathbf{K}_{22} - \mathbf{K}_{21}(\mathbf{K}_{11} + m\lambda \mathbf{I})^{-1}\mathbf{K}_{21}'.
\end{aligned}$$

The variance covariance matrix $\hat{\gamma}$ for proportional to

$$(\mathbf{K}_{22} - \mathbf{K}_{21}(\mathbf{K}_{11} + \lambda I)^{-1}\mathbf{K}_{12}). \quad (3)$$

Reproducing Kernel Hilbert Spaces (RKHS) regression methods replace the inner products by kernels, it is as if we are performing ridge regression on a transformed data $\phi(x)$, where ϕ is a feature map associated to the chosen kernel function and the associated kernel matrix. The resulting predictor is now nonlinear in x and agrees with the predictor derived from the RKHS perspective (Schölkopf and Smola 2002). RKHS regression extends ridge regression allowing a wide variety of kernel matrices, not necessarily additive in the input variables, calculated using a variety of kernel functions. A kernel function, $k(., .)$ maps a pair of input points \mathbf{x} and \mathbf{x}' into real numbers. It is by definition symmetric ($k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$) and non-negative. Given the inputs for the n individuals we can compute a kernel matrix K whose entries are $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. The common choices for kernel functions are the linear ($k(\mathbf{x}; \mathbf{y}) = \mathbf{x}'\mathbf{y}$), polynomial ($k(\mathbf{x}; \mathbf{y}) = (\mathbf{x}'\mathbf{y} + c)^d$ for c and $d \in R$), Gaussian kernel functions ($k(\mathbf{x}; \mathbf{y}) = \exp(-h(\mathbf{x}' - \mathbf{y}')(\mathbf{x}' - \mathbf{y}'))$ where $h > 0$), though many other options are available (Schölkopf and Smola 2002). Reproducing Kernel Hilbert Spaces Regressions (RKHS) have been used for regression (e.g., Smoothing Spline Wahba 1990), spatial smoothing (e.g., Kriging Cressie 1988) and classification problems (e.g., Support Vector Machine, Vapnik 1998). Gianola *et al.* (2006), proposed to use this approach for genomic prediction and, since then several follow-up articles with focus on the application of these methods to various genome-wide regression problems have also been published (González-Recio *et al.* 2008).

Gaussian Linear Mixed Models

The linear mixed model methodology was first developed within the context of animal genetics and breeding research by Henderson (1975); Kempthorne *et al.* (1957); Henderson *et al.* (1959), many important statistical models can be expressed as mixed effects models and it is the most widely used model in prediction of quantitative traits, and genome-wide association studies. In studies on linear mixed models it is usual to consider the estimation of the fixed effects β and the variance components, and also the prediction of the random effects \mathbf{u} . For a given

data vector \mathbf{y} , the vector of random effects \mathbf{u} is a realization of random variables which are observed and these effects must therefore necessarily be predicted from the data (Henderson 1953).

In the linear mixed-effects model, the observations are assumed to result from a hierarchical linear model:

$$\mathbf{y} = \mathbf{W}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon};$$

and $\boldsymbol{\epsilon} \sim N(0, R)$ is independent of $\mathbf{u} \sim N(0; G)$. These assumptions imply $E(\mathbf{y}|\mathbf{W}; \mathbf{Z}) = \mathbf{W}\boldsymbol{\beta}$, $\mathbf{y} \sim N(\mathbf{W}\boldsymbol{\beta}; \mathbf{ZGZ}' + R) = N(\mathbf{W}\boldsymbol{\beta}; V)$.

The similarity of the mixed models and RKHS regression models has been stressed many times. However, mixed modeling approach provides a formalized approach since the inferences are based on a probabilistic model, and therefore, allows legitimate inferences about the parameters and predictions.

Henderson et al. show that maximizing the joint density of \mathbf{y} and \mathbf{u} yields the MLEs of the parameters $\boldsymbol{\beta}$ and EBLUPs (estimated BLUPs) $\hat{\mathbf{u}}$ that solve: $\mathbf{W}'R^{-1}\mathbf{W}\hat{\boldsymbol{\beta}} + \mathbf{W}'R^{-1}\mathbf{Z}\hat{\mathbf{u}} = \mathbf{W}'R^{-1}\mathbf{y}$ and $\mathbf{Z}'R^{-1}\mathbf{W}\hat{\boldsymbol{\beta}} + \mathbf{Z}'R^{-1}\mathbf{Z}\hat{\mathbf{u}} + G^{-1}\hat{\mathbf{u}} = \mathbf{Z}'R^{-1}\mathbf{y}$, this leads to the Henderson's mixed model equations:

Henderson's mixed-model equations can be used to estimate the standard errors of the fixed and random effects. For a given design, the inverse of the coefficient matrix is written as

$$\begin{bmatrix} \mathbf{W}'R^{-1}\mathbf{W} & \mathbf{W}'R^{-1}\mathbf{Z} \\ \mathbf{Z}'R^{-1}\mathbf{W} & \mathbf{Z}'R^{-1}\mathbf{Z} + G^{-1} \end{bmatrix}^{-1} = \begin{bmatrix} H_{11} & H_{12} \\ H'_{12} & H_{22} \end{bmatrix}$$

where H_{11} , H_{12} , and H_{22} are, respectively, $p \times p$, $p \times q$, and $q \times q$ sub-matrices. Note that referring to the coefficient matrix is an abuse of notation since the parameters of the mixed effects model does not include the vector \mathbf{u} . Using this notation, the sampling covariance matrix for the BLUE (best linear unbiased estimator) of $\boldsymbol{\beta}$ is given by $\sigma(\boldsymbol{\beta}) = H_{11} = (\mathbf{W}'V^{-1}\mathbf{W})^{-1}$ that the sampling covariance matrix of the prediction errors $(\hat{\mathbf{u}} - \mathbf{u})$ is given by

$$\text{cov}(\hat{\mathbf{u}} - \mathbf{u}) = H_{22} = G - GZ'PZG \quad (4)$$

for $P = V^{-1} - V^{-1}\mathbf{W}(\mathbf{W}'V^{-1}\mathbf{W})^{-1}\mathbf{W}'V^{-1}$ and that the sampling covariance of estimated effects and prediction errors is given by $\sigma(\boldsymbol{\beta}, \hat{\mathbf{u}} - \mathbf{u}) = H_{12} = -(\mathbf{W}'V^{-1}\mathbf{W})^{-1}\mathbf{W}'V^{-1}\mathbf{Z}G$ (We consider $\hat{\mathbf{u}} - \mathbf{u}$ rather than $\hat{\mathbf{u}}$ as the latter includes variance from both the prediction error and the random effects \mathbf{u} themselves.). The standard errors of the fixed and random effects are obtained, respectively, as the square roots of the diagonal elements of H_{11} and H_{22} . In addition, using the above definitions, $\text{cov}(\mathbf{u}|\mathbf{y}) = G - GZ'V^{-1}\mathbf{Z}G = (\mathbf{Z}'R^{-1}\mathbf{Z} + G^{-1})^{-1}$.

Optimal design of experiments with mixed models involve determination of the design matrices \mathbf{W} and \mathbf{Z} ; however, in many applications, estimates of only one of $\boldsymbol{\beta}$ or \mathbf{u} is needed. For

example, design criterion is obtained by considering the variance-covariance matrix of $C'(\hat{\mathbf{u}} - \mathbf{u})$ given by $C'H_{22}C$ is named the prediction error variance. A more recent design criterion is the generalized coefficient of determination (Laloë 1993; Laloë and Phocas 2003; Rincent *et al.* 2012) for the random terms $\mathbf{c}_i'(\hat{\mathbf{u}} - \mathbf{u})$, $i = 1, \dots, l$:

$$\sum_{i=1}^l \frac{\mathbf{c}_i' H_{22} \mathbf{c}_i}{\mathbf{c}_i' G \mathbf{c}_i}$$

for a set of contrasts \mathbf{c}_i .

If the mixed model is simplified such that $\boldsymbol{\epsilon} \sim N(0, R = \sigma_{\boldsymbol{\epsilon}}^2 I)$ and $\mathbf{u} \sim N(0; G = \sigma_{\mathbf{u}}^2 A)$, and the rows of C have zeros corresponding to fixed effects, the formula for prediction error variance becomes:

$$C(Z'MZ + \lambda A^{-1})^{-1}C'$$

and the corresponding formula for coefficient of determination becomes:

$$\sum_{i=1}^l \frac{\mathbf{c}_i'(A - \lambda(Z'MZ + \lambda A^{-1})^{-1})\mathbf{c}_i}{\mathbf{c}_i' A \mathbf{c}_i}, \quad (5)$$

where $\lambda = \sigma_{\boldsymbol{\epsilon}}^2 / \sigma_{\mathbf{u}}^2$. Furthermore, when we assume $\mathbf{u} \sim N(\mathbf{0}, G = \sigma_{\mathbf{u}}^2 I)$, then the above formulas simplify further to

$$C(Z'MZ + \lambda I)^{-1}C'$$

and

$$\sum_{i=1}^l \left(1 - \frac{\lambda \mathbf{c}_i'(Z'MZ + \lambda I)^{-1} \mathbf{c}_i}{\mathbf{c}_i' \mathbf{c}_i}\right).$$

Here, $M = I - W(W'W)^{-1}W'$ is a projection matrix orthogonal to the vector subspace spanned by the columns of W , so that $MW = 0$.

Some generalizations and extensions of parametric design criteria

A generalization of the D -, A -, G - optimal criteria is provided by the Keifer's ϕ_p criteria: given by

$$\phi_p(\zeta) = (\text{trace}(I_{\boldsymbol{\theta}}(\zeta))^{-p})^{1/p},$$

where $-1 \leq p < \infty$. $p = 0, 1, \infty$ gives the D -, A -, E - optimal criteria correspondingly.

Another extension of D -optimality deals with minimizing

$$\sum_{j=1}^h \alpha_j \log |A_j I(\zeta)^{-1} A_j'|.$$

The criterion permits designs for h different models which may be fitted to the data, for the

j th of which the information matrix is $I_j(\zeta)$. The matrix A_j defines S_j linear combinations of the p_j parameters in model j which are of experimental importance and the non-negative weights α_j express the relative importance of the different aspects of the design. Examples of compound D -optimum designs for linear models are given by [Atkinson and Donev \(1992\)](#).

[Pritchard and Bacon \(1978\)](#) proposed a new criterion alternative to the traditional D -optimal design, which has a measure of the overall correlation among the parameters directly as objective function to be minimized i.e. the root square of the individual correlations between pair of parameters:

$$F = \left(\sum_{i,j} \frac{corr_{ij}^2}{p^2 - p} \right)^{1/2}.$$

Non-parametric design criteria

The design criteria of the previous sections started from a parametric model. There are some optimal design approaches that does not make any parametric assumptions, leading to non-parametric design criteria. Most of these methods are based on a distance matrix.

A design criteria that aims to achieve a high spread among its support points within the design region, i.e., make the smallest distance between neighboring points in the design as large as possible is called the maximin-distance criterion. Let $D = d_{ij, i=1, \dots, N}$ denote the distance matrix among the possible design points. Maximin distance criteria is finding the subset of n points such that

$$\phi_1(\zeta) = \min(d_{ij}), i \neq j$$

to be maximized among these n points. Another possibility is to pick the n design points so that the the maximum distance from all the points in a target set of points X^* is as small as possible. These designs are called space filling designs, some performance bench-marking for various space-filling designs can be found in ([Pronzato 2008](#)) and ([Pronzato and Müller 2012](#)).

Example 1: In this example, we want to find the best D -optimal 13 point design for a second order regression model over a grid design region defined by two variables both with possible values in the set $-2, -1, 0, 1, 2$. Naming these variables as x_1 and x_2 and the generic response as y , we can write this model as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 + \epsilon.$$

First, we crate the design matrix for the design space.

Box 1: Creating the design matrix for grid, selecting "best" subset by enumeration and using STPGA

```

> library(STPGA)
> set.seed(1234)
> X<-matrix(0,nrow=5^2,ncol=5)
> ij=0
> for (i in -2:2){
+   for (j in -2:2){
+     ij=ij+1
+     X[ij,]<-c(i,j, i^2,j^2, i*j)
+   }
+ }
> X<-cbind(1,X)
> rownames(X)<-paste("x",1:5^2, sep="")
> #lisofallsubsetsofsize13<-combn(rownames(X), 13)
> #dim(lisofallsubsetsofsize13)
> #####complete enumeration of
> #####(5^2 choose 13)=5200300 possibilities
> #I have done this once, you dont need to do it.
> #DOPTVALS<-apply(lisofallsubsetsofsize13, 2,
+   function(x){DOPT(Train=x, Test=NULL, P=X, lambda = 1e-09, C=NULL)})
> BESTSOL<-c("x1", "x2", "x3", "x5", "x6", "x10",
+   "x11", "x13", "x15", "x21", "x22", "x24", "x25")
> #BESTSOL<-lisofallsubsetsofsize13[,which.min(DOPTVALS)]
> #best solution is not unique for this problem
> mindoptvals<--21.3096195830339709687
> #mindoptvals<-min(DOPTVALS)
> ListTrain1<-GenAlgForSubsetSelectionNoTest(P=X,ntoselect=13, InitPop=NULL,
+   npop=200, nelite=5, mutprob=.5, mutintensity = 1,
+   niterations=200,minitbefstop=50, tabu=FALSE,
+   tabumemsize = 0,plotiters=FALSE,
+   lambda=1e-9,errorstat="DOPT", mc.cores=4)
> length(intersect(ListTrain1$`Solution with rank 1`,BESTSOL))

[1] 12

> mindoptvals==min(ListTrain1$`Best criterion values over iterarions`)

[1] TRUE

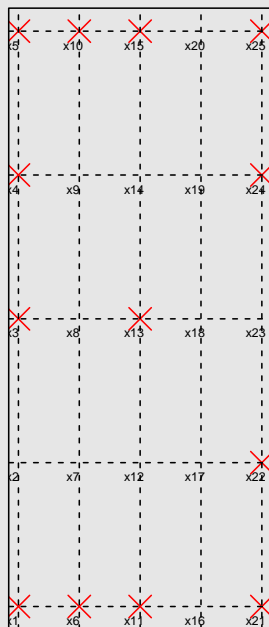
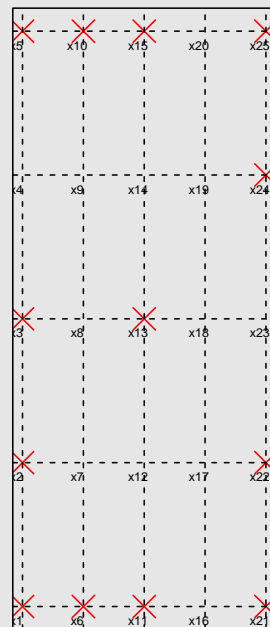
```

Box 2: Plotting the results on the grid

```

> par(mfrow=c(1,2))
> labelling1<-rownames(X)%in%ListTrain1$`Solution with rank 1`+1
> plot(X[,2], X[,3], col=labelling1,
+      pch=2*labelling1,cex=2*(labelling1-1),
+      xlab="", ylab="", main="STPGA solution",
+      cex.main=.7,xaxt='n',yaxt='n')
> text(x=X[,2]-.1, y = X[,3]-.1, labels = rownames(X), cex=.5)
> for (i in -2:2){
+   abline(v=i, lty=2)
+   abline(h=i,lty=2)
+ }
> labelling2<-rownames(X)%in%BESTSOL+1
> plot(X[,2], X[,3], col=labelling2,
+      pch=2*labelling2,cex=2*(labelling2-1),
+      xlab="", ylab="", main="Best solution",
+      cex.main=.7,xaxt='n',yaxt='n')
> text(x=X[,2]-.1, y = X[,3]-.1, labels = rownames(X), cex=.5)
> for (i in -2:2){
+   abline(v=i, lty=2)
+   abline(h=i,lty=2)
+ }
> par(mfrow=c(1,1))

```

STPGA solution**Best solution**

Example 2: In statistical genetics, an important task involves building predictive models of the genotype-phenotype relationship to be able to make genomic predictions and also to attribute a proportion of the total phenotypic variance to locations on the genome (genomewide association studies (GWAS)). If the genotypic information for the candidates (and the target) are available, phenotypic experiments can be executed for a subset of these individuals that is optimal according to a design criteria that only uses the available genotypic information.

STPGA package comes with a genomic data set called **WheatData** involving phenotypes and markers for 200 elite wheat lines selected at random from a larger genetic pool. Data was downloaded from the website triticeaetoolbox.org. The 4670 markers available for these 200 genotypes were preprocessed for missingness and minor allele frequencies, coded numerically as 0, 1, and 2; the relationship genomic relationship matrix was calculated according to the formula in Van Raden (VanRaden 2008): $\frac{M_c M_c'}{k}$ where $k = \sum_{j=1}^m 2p_j(1 - p_j)$ is twice the sum of heterozygosities of the markers and M_c the allele counts matrix M centered by the mean frequencies of alleles. The genotypic values for plant heights were predicted using a mixed effects model that is fitted to a multi-environmental trial involving these genotypes and the corresponding phenotypic observations.

As long as the model assumptions are correct for the data and a suitable criterion is employed, the prediction accuracies of models built on optimal sets are expected to be better than average prediction accuracies of models based on a random set of the same size. In addition, if a target set is specified, further gains might be achieved using this knowledge. GWAS results based on a genetic information based optimal design is expected to improve the association results compared to models built on phenotypic experiments performed on a random set of the same size in a similar way. Obtaining genotypic information is becoming cheaper by the day, however the high costs and challenges related to phenotypic experiments persist. To see this is the case consider what might be involved in a longitudinal study on human subjects.

Optimal design of phenotypic experiments based on prior genotypic information can lead to high information value at low costs. To illustrate these points, let's use the wheat data set first in prediction and association settings. We begin by loading the data and doing some preprocessing necessary to run the experiment:

Box 3: Loading and preprocessing the wheat data set included in STPGA

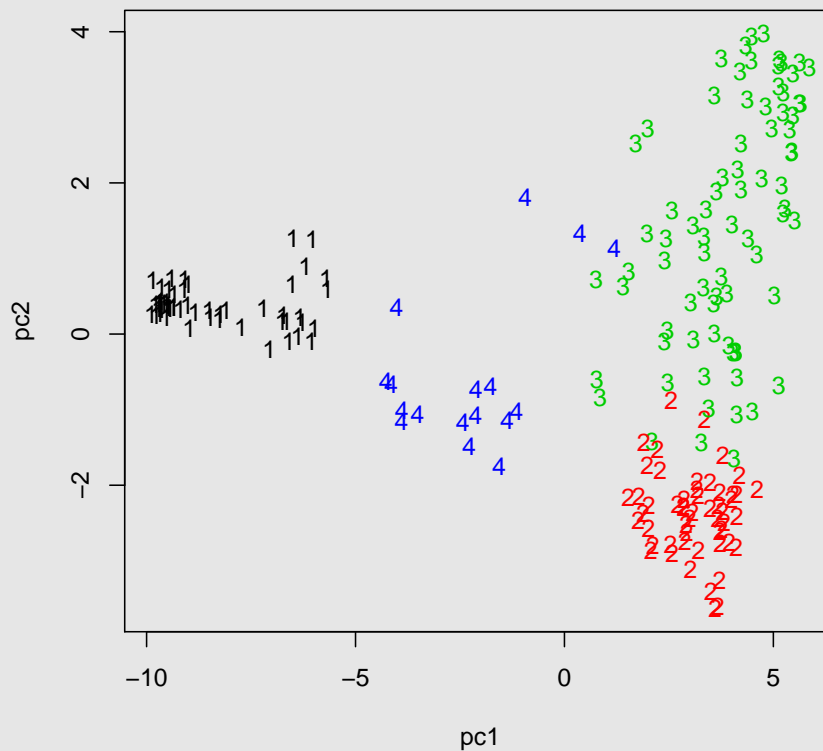
```
> data(WheatData)
> svdWheat<-svd(Wheat.K, nu=50, nv=50)
> PC50Wheat<-Wheat.K%*%svdWheat$v
> rownames(PC50Wheat)<-rownames(Wheat.K)
> DistWheat<-dist(PC50Wheat)
> TreeWheat<-cutree(hclust(DistWheat), k=4)
```

"TreeWheat" partitions the data into four sets, let's observe this grouping using a plot of first

two principal components.

Box 4: Plotting the observations in wheat data using first two principal components

```
> plot(PC50Wheat[,1],PC50Wheat[,2], col=TreeWheat,
+      pch=as.character(TreeWheat), xlab="pc1", ylab="pc2")
```



We will consider a scenario where the final aim is to accurately predict the genotypic values in group 2 and we want to establish this by conducting a phenotypic experiment on 50 genotypes selected from the remaining groups.

Box 5: Splitting the data into Candidates and Test

```

> Test<-rownames(PC50WHeat)[TreeWheat==2]
> length(Test)

[1] 53

> Candidates<-setdiff(rownames(PC50WHeat), Test)
> Ztrainfull<-as.matrix(model.matrix(~1+factor(Candidates,
+           levels=rownames(Wheat.M))))
> deptimefull<-Wheat.Y[Wheat.Y$id%in%Candidates,]
> dim(deptimefull)

[1] 147  2

```

Once the data is ready, it is easy to call STPGA with the default options. Note that there are two options, we can use the information about the target genotypes or we could ignore this. I will do both:

Box 6: Optimization of training populations with STPGA

```

> Train1<-GenAlgForSubsetSelection(P=PC50WHeat,Candidates=Candidates,
+   Test=Test, ntoselect=50, mc.cores=4)
> Train2<-GenAlgForSubsetSelectionNoTest(P=PC50WHeat,
+   ntoselect=50, mc.cores=4)

```

However, it is important to be able to specify GA parameters:

Box 7: Optimization of training populations with STPGA, specifying algorithm parameters

```

> Train3<-GenAlgForSubsetSelection(P=PC50WHeat,Candidates=Candidates,
+   Test=Test,ntoselect=50,
+   InitPop=NULL,npop=200,
+   nelite=10, mutprob=.5, mutintensity = 1,niterations=200,
+   minitbefstop=50, tabumemsize = 1,plotiters=FALSE,
+   lambda=1e-9,errorstat="PEVMEAN", mc.cores=4)
> Train4<-GenAlgForSubsetSelectionNoTest(
+   P=PC50WHeat[rownames(PC50WHeat)%in%Candidates,],ntoselect=50,
+   InitPop=NULL,npop=200,
+   nelite=10, mutprob=.5, mutintensity = 1,niterations=200,
+   minitbefstop=50, tabumemsize = 1,plotiters=FALSE,
+   lambda=1e-9,errorstat="PEVMEAN", mc.cores=4)

```

We finally want to compare the prediction accuracy for predicting the target data compared to the average accuracy that would be obtained using a sample size of same size. I will only use "Train3" and "Train4" below, we will also need the the package **EMMREML** (Akdemir and Godfrey 2015) for fitting the mixed effects model:

Box 8: Building models based on optimal samples, getting predictions

```

> require("EMMREML")
> deptest<-Wheat.Y[Wheat.Y$id%in%Test,]
> Ztest<-model.matrix(~-1+deptest$id)
> ##predictions by optimized sample
> deptrainopt3<-Wheat.Y[(Wheat.Y$id%in%Train3$`Solution with rank 1`),]
> Ztrain3<-model.matrix(~-1+deptrainopt3$id)
> modelopt3<-emmreml(y=deptrainopt3$plant.height,
+                   X=matrix(1, nrow=nrow(deptrainopt3), ncol=1),
+                   Z=Ztrain3, K=Wheat.K)
> predictopt3<-Ztest%*%modelopt3$uhat
> #####
> deptrainopt4<-Wheat.Y[(Wheat.Y$id%in%Train4$`Solution with rank 1`),]
> Ztrain4<-model.matrix(~-1+deptrainopt4$id)
> modelopt4<-emmreml(y=deptrainopt4$plant.height,
+                   X=matrix(1, nrow=nrow(deptrainopt4), ncol=1),
+                   Z=Ztrain4, K=Wheat.K)
> predictopt4<-Ztest%*%modelopt4$uhat

```

We will repeat estimation with random sample 300 times to obtain mean performance:

Box 9: Estimating the accuracy of a random sample of the same size

```

> corvecrs<-c()
> for (rep in 1:300){
+   rs<-sample(Candidates, 50)
+   +
+   deptrainrs<-Wheat.Y[(Wheat.Y$id%in%rs),]
+   +
+   Ztrainrs<-model.matrix(~-1+deptrainrs$id)
+   +
+   modelrs<-emmreml(y=deptrainrs$plant.height,
+                   X=matrix(1, nrow=nrow(deptrainrs), ncol=1),
+                   Z=Ztrainrs, K=Wheat.K)
+   predictrs<-Ztest%*%modelrs$uhat
+   corvecrs<-c(corvecrs,cor(predictrs, deptest$plant.height))
+ }

```

Here are the results:

Box 10: Comparisons of accuracies

```
> ##average accuracy random sample
> mean(corvecrs)

[1] 0.303162

> ##accuracy of Train3$`Solution with rank 1`
> cor(predictopt3, deptest$plant.height)

      [,1]
[1,] 0.3146401

> ##accuracy of Train3$`Solution with rank 1`
> cor(predictopt4, deptest$plant.height)

      [,1]
[1,] 0.3936563
```

These results are as expected: Optimally designed phenotypic experiments are more informative, they result in higher accuracies compared to a random sample of the same size. If the researcher also has access to the genotypic information for the individuals in the target set, then this information when properly used might lead to gains in per unit information that will come from a phenotypic experiment.

I also expect that the association (GWAS) results from an optimized sample to be better than a random sample. I can not verify this with a simple example. However, here is a comparison of the marker effects estimated from a full set, compared to a random sample and an optimized sample of the same size.

Box 11: Using STPGA in training population selection for GWA studies

```

> modelrrblupfull<-emmreml(y=deptrainfull$plant.height,
+                           X=matrix(1, nrow=nrow(deptrainfull), ncol=1),
+                           Z=Ztrainfull%*%Wheat.M, K=diag(ncol(Wheat.M)))
> Trainopt<-GenAlgForSubsetSelectionNoTest(
+   P=PC50Wheat[rownames(PC50Wheat)%in%Candidates,], ntoselect=50,
+   InitPop=NULL, npop=200,
+   nelite=10, mutprob=.5, mutintensity = 1, niterations=100,
+   minitbefstop=50, tabumemsize = 1, plotiters=FALSE,
+   lambda=1e-9, errorstat="DOPT", mc.cores=4)
> deptrainopt<-Wheat.Y[(Wheat.Y$id%in%Trainopt$`Solution with rank 1`),]
> Ztrainopt<-model.matrix(~1+deptrainopt$id)
> modelrrblupopt<-emmreml(y=deptrainopt$plant.height,
+                           X=matrix(1, nrow=nrow(deptrainopt), ncol=1),
+                           Z=Ztrainopt%*%Wheat.M, K=diag(ncol(Wheat.M)))
> modelrrbluprs<-emmreml(y=deptrainopt4$plant.height,
+                           X=matrix(1, nrow=nrow(deptrainopt4), ncol=1),
+                           Z=Ztrainrs%*%Wheat.M, K=diag(ncol(Wheat.M)))
> orderfull<-order(abs(modelrrblupfull$uhat), decreasing=T)
> orderopt<-order(abs(modelrrblupopt$uhat), decreasing=T)
> orderrrs<-order(abs(modelrrbluprs$uhat), decreasing=T)
> mean(abs(orderrrs-orderfull))

[1] 1580.187

> mean(abs(orderopt-orderfull))

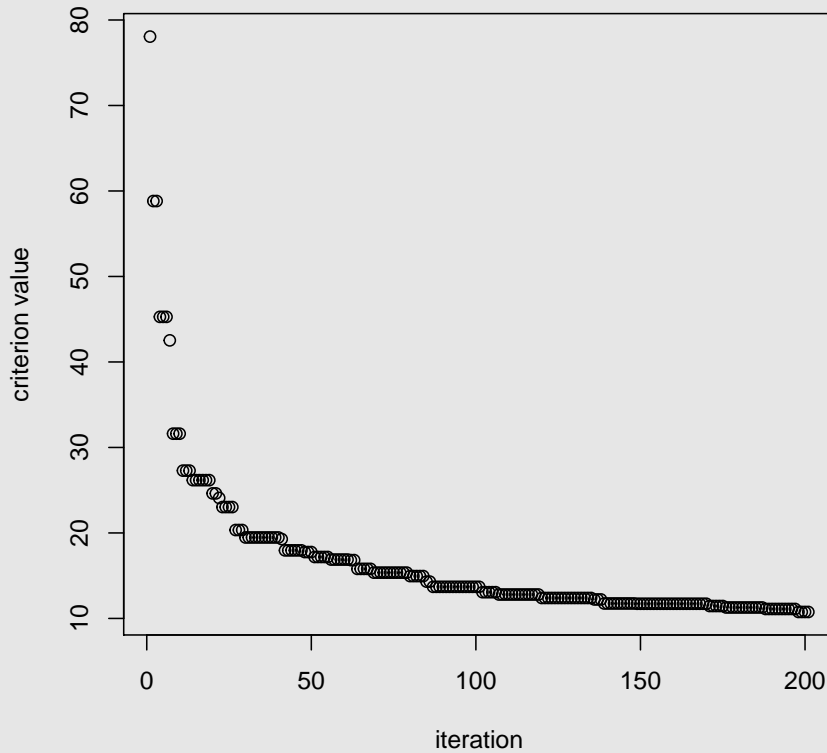
[1] 1567.686

```

As noted before, the subset selection optimization problem is a combinatorial one. We need to see if the algorithm got close to convergence, we can do this by plotting the criterion values over the iterations, these values are stored in the output of STPGA with the name ‘Best criterion values over iterations’.

Box 12: Plotting the progress of the optimization

```
> plot(Train3$`Best criterion values over iterarions`,
+      xlab="iteration", ylab="criterion value")
```



The users are recommended to check the convergence of the algorithm by checking the plot, the last iterations should show little or no improvement. If the algorithm has still room to improve the solutions from the last run can be used as new starting points and the iteration can be restarted, perhaps with different settings for the GA parameters. For difficult problems, a good strategy is to run the algorithm several times and select the best solution among these. It is also possible to implement the genetic algorithm in an island model, I have included the code for a simple island model in **Appendix**.

4.2. STPGA in other subset selection problems

Optimal subset selection algorithm in STPGA can be used with user supplied optimization criterion, and therefore, it has a wide area of application. I am going to try to give a few examples: supervised unsupervised variable selection, Minimize inbreeding while maximizing gain, mixed integer programming, influential observation selection. These examples can easily

be extended.

The following example illustrates how the users can define their own criteria and use it with STPGA for a variable selection problem. It involves aligning kernels to select variables and as far as I know this wasn't done before.

Selecting most representative marker set (markers that represent most of the variability in a given marker data)

A genetic relationship matrix measures the amount individuals in a certain group are genetically similar. Genetic relationship matrices can be constructed using pedigrees, or using genome-wide markers. In this example, we will try to find a fixed size subset of available genome-wide markers that results in a genetic relationship matrix that is as close to the genetic relationship matrix as possible. These selected markers can be called the *genetic anchor markers*, since they explain most of the properties of the genome-wide relationship matrix. The main question is if there is a subset of markers that can explain a big part of all of the variation captured by all the markers (or even the genome sequence), since this relates to many important genetic concepts like effective population size, effective number of independent chromosome segments, population structure and its effects on predictability within and between sub-populations. Note that the selection of the anchor markers does not involve any phenotypic observations, therefore this is an unsupervised marker selection approach, similar to some recent approaches expressed in sparse principal components analysis ([Witten et al. 2009](#)) or sparse partial least squares ([Chun and Keleş 2010](#)). However, the interpretation of the factors extracted by sparse PCA and sparse PLS might be difficult since these are linear combinations of the original variables.

The following is a simple function for obtaining a genetic relationship matrix given the matrix of markers (n x m) (n:number of genotypes), (m:number of markers) coded as 0, 1, 2 representing the number of minor alleles. A function to calculate the relationship genomic relationship matrix according to the formula in Van Raden ([VanRaden 2008](#)) is given below:

Box 13: A function to calculate Van Raden's relationship matrix from minor allele frequency scores (markers coded as 0,1 and 2)

```
> A.mat<-function(M){
+ pks<-colMeans(M)/2
+ W<-scale(M, center=TRUE, scale=FALSE)
+ c<-2*sum(pks*(1-pks))
+ Amat<-tcrossprod(W)/c
+ rownames(Amat)<-colnames(Amat)<-rownames(M)
+ return(Amat)
+ }
```

I will only use the lines in the second cluster, the whole data would take too much time to process.

Box 14: Obtaining a subset of wheat data set for a quick analysis

```

> #the problem can be solved for all the genotypes to make the
> #problem computationally easier, we will pick only the
> #genotypes in forth cluster
> library(Matrix)
> Wheat.M4<-Wheat.M[rownames(Wheat.M)%in%names(TreeWheat[TreeWheat==2]),]
> Wheat.M4<-Matrix(Wheat.M4+1)
> #relationship using all markers
> ##A.mat requires the markers are coded as 0, 1, 2
> Afull<-A.mat(M=Wheat.M4)

```

We can see how this optimally selected markers compare with the randomly selected marker sets of the same size.

Box 15: Optimally selected anchor markers versus randomly selected markers

```

> n<-nrow(Wheat.M4)
> diffvecrs<-c()
> for (i in 1:100){
+ rsmallM<-Wheat.M4[,sample(1:ncol(Wheat.M4),50)]
+ Ars<-A.mat(M=rsmallM)
+ diffvecrs<-c(diffvecrs,mean((c(Afull[lower.tri(Afull, diag=TRUE)])
+                               -c(Ars[lower.tri(Ars, diag=TRUE)]))^2))
+ }
> #User defined criterion
> STPGAUSERDEFFUNC<-function(Train,Test=NULL, P, lambda=1e-6, C=NULL){
+ trsmallM<-t(P[rownames(P)%in%Train,])
+ Atr<-A.mat(M=trsmallM)
+ return(mean((c(Afull[lower.tri(Afull, diag=TRUE)])
+               -c(Atr[lower.tri(Atr, diag=TRUE)]))^2))
+ }
> GAOUT<-GenAlgForSubsetSelectionNoTest(P=t(Wheat.M4),
+   ntoselect=50,npop=300,
+   nelite=10, mutprob=.5, mutintensity = 1,
+   niterations=400, minitbefstop=50,tabu=FALSE,
+   tabumemsize = 1,plotiters=F,lambda=1e-6,
+   errorstat="STPGAUSERDEFFUNC",mc.cores=4)
> min(GAOUT$`Best criterion values over iterarions`);mean(diffvecrs)

[1] 0.03310715

[1] 0.1044086

> optsmallM<-Wheat.M4[, colnames(Wheat.M4)%in%GAOUT$`Solution with rank 1`]
> optA<-A.mat(optsmallM)

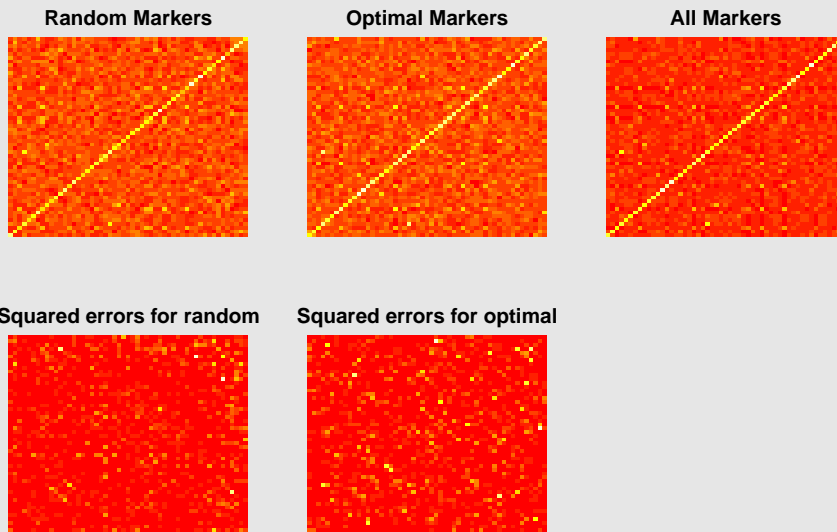
```

Box 16: Plotting the results for optimally selected "anchor markers" versus randomly selected markers

```

> layout(matrix(c(1,2,3,4,5,6),2,3, byrow=TRUE), widths=c(2,2,2),
+         heights=c(2,2), respect=TRUE)
> par(mar=c(3,2,2,1))
> # turn off the axes
> image(Ars, axes=FALSE, main="Random Markers")
> image(optA, axes=FALSE, main="Optimal Markers")
> image(Afull, axes=FALSE, main="All Markers")
> image((Ars-Afull)^2, axes=FALSE,
+       main="Squared errors for random")
> image((optA-Afull)^2, axes=FALSE,
+       main="Squared errors for optimal")
> par(mfrow=c(1,1))

```



According to these results, the optimally selected markers sets result in a genetic relationship matrix that is closer to the full marker genetic relationship matrix than the relationship matrices calculated from random sets of markers. I also want to see how the variance captured by these relationship matrices and the accuracies of the models built using these relationship matrices compare.

Box 17: Comparing accuracy of predictions for optimally selected anchor markers, randomly selected markers and all markers

```

> linenames<-rownames(Afull)
> Test<-sample(linenames, 20)
> Train<-setdiff(linenames, Test)
> Wheat.Y4<-Wheat.Y[Wheat.Y$id%in%rownames(Afull),]
> Wheat.Y4$id<-factor(as.character(Wheat.Y4$id), levels=linenames)
> Wheat.Y4Train<-Wheat.Y4[Wheat.Y4$id%in%Train,]
> Wheat.Y4Test<-Wheat.Y4[Wheat.Y4$id%in%Train,]
> Ztrain<-model.matrix(~-1+Wheat.Y4Train$id)
> Ztest<-model.matrix(~-1+Wheat.Y4Test$id)
> library(EMMREML)
> modelfull<-emmreml(y=Wheat.Y4Train$plant.height,
+                   X=matrix(rep(1,nrow(Ztrain)), ncol=1),
+                   Z=Ztrain, K=Afull+1e-9*diag(nrow(Afull)))
> modelfull$Vu

[1] 63.14514

> rsVus<-c()
> for( i in 1:100){
+   rssmallM<-Wheat.M4[,sample(1:ncol(Wheat.M4),50)]
+   Ars<-A.mat(M=rssmallM)
+   modelrs<-emmreml(y=Wheat.Y4Train$plant.height,
+                   X=matrix(rep(1,nrow(Ztrain)), ncol=1),
+                   Z=Ztrain, K=Ars+1e-9*diag(nrow(Ars)))
+   rsVus<-c(rsVus,modelrs$Vu)
+ }
> mean(rsVus)

[1] 32.18926

> modeloptA<-emmreml(y=Wheat.Y4Train$plant.height,
+                   X=matrix(rep(1,nrow(Ztrain)), ncol=1),
+                   Z=Ztrain, K=optA+1e-9*diag(nrow(optA)))
> modeloptA$Vu

[1] 30.73143

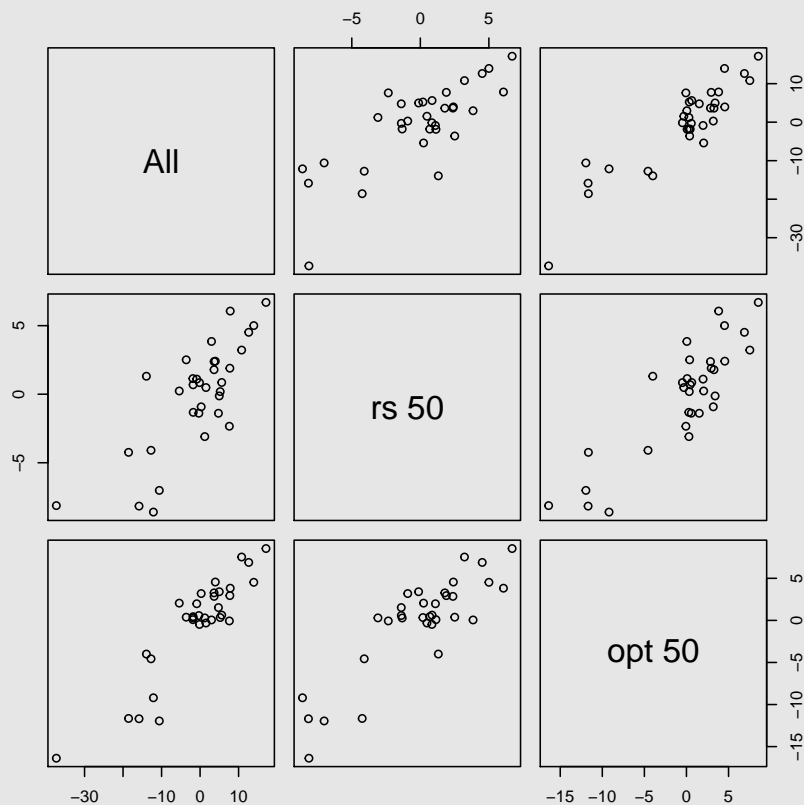
> predictmatrix<-as.matrix(Ztest%*%cbind(modelfull$uhat,
+   modelrs$uhat,modeloptA$uhat))
> colnames(predictmatrix)<-c("All", "rs 50", "opt 50")

```

Box 18: Comparing accuracy of predictions for optimally selected anchor markers, randomly selected markers and all markers

```
> pairs(predictmatrix)
> cor(predictmatrix)
```

	All	rs 50	opt 50
All	1.0000000	0.7793880	0.9094115
rs 50	0.7793880	1.0000000	0.8656134
opt 50	0.9094115	0.8656134	1.0000000



Minimize inbreeding while maximizing gain

Many authors (Goddard 2009; Jannink 2010; Sun *et al.* 2013; Akdemir and Sánchez 2016) have expressed the importance of reducing inbreeding in PS and GS for long-term success of breeding programs. They argued that GS is likely to lead to a more rapid decline in the selection response unless new alleles are continuously added to the calculation of GEBVs, stressing the importance of balancing short and long term gains by controlling inbreeding in selection.

A quadratic programming (QP) problem has the generic form

$$Q(\mathbf{x}) = \frac{1}{2}\mathbf{x}'D\mathbf{x} - \mathbf{d}'\mathbf{x} + c.$$

Here, \mathbf{x} is a vector in \mathbb{R}^n , D is an $n \times n$ symmetric positive definite matrix, \mathbf{d} is a constant vector in \mathbb{R}^n and c is a scalar constant. QPs usually come with a system of linear constraints on the vector $\mathbf{x} \in \mathbb{R}^n$ which can be written as

$$A\mathbf{x} = \mathbf{f} \quad B\mathbf{x} \geq \mathbf{g}.$$

Here A is an $m_1 \times n$ matrix with $m_1 \leq n$ and B is a $m_2 \times n$ matrix. The vectors \mathbf{f} and \mathbf{g} have lengths m_1 and m_2 respectively. QP can be more compactly stated as compactly as:

$$\begin{cases} \text{minimize}_{\mathbf{x} \in \mathbb{R}^n} : & Q(\mathbf{x}) = \frac{1}{2}\mathbf{x}'D\mathbf{x} - \mathbf{d}'\mathbf{x} + c \\ \text{subject to} : & A\mathbf{x} = \mathbf{f} \quad B\mathbf{x} \geq \mathbf{g} \end{cases}$$

There are many efficient algorithms that solves QP's so there is in practice little difficulty in calculating the optimal solution for any particular data set. In this example, I will be using the package **quadprog** (Turlach and Weingessel 2007).

Now, let A be a matrix of pedigree based numerator relationships or the additive genetic relationships between the individuals in a breeding population (this matrix can be obtained from a pedigree of genome-wide markers for the individuals) and let \mathbf{c} be the vector of proportional contributions of individuals to the next generation under a random mating scheme. The average inbreeding and co-ancestry for a given choice of \mathbf{c} can be defined as $r = \frac{1}{2}\mathbf{c}'A\mathbf{c}$. If \mathbf{b} is the vector of GEBV's, i.e., the vector of BLUP (best linear unbiased predictor) estimated BV's of the candidates for selection. The expected gain is defined as $g = \mathbf{c}'\mathbf{b}$. Without loss of generality, we will assume that the breeder's long term goal is to increase the value of g .

In Wray and Goddard (1994); Brisbane and Gibson (1995); Meuwissen (1997), an approach that seeks minimizing the average inbreeding and co-ancestry while restricting the genetic gain is proposed. The optimization problem can be stated as

$$\begin{aligned} \text{minimize}_{\mathbf{c}} \quad & r = \mathbf{c}'\frac{A}{2}\mathbf{c} \\ \text{subject to} \quad & \mathbf{c}'\mathbf{b} = \rho \\ & \mathbf{c}'\mathbf{1} = 1 \\ & \mathbf{c} \geq 0, \end{aligned} \tag{6}$$

where ρ is the desired level of gain.

This problem is easily recognized as a QP. Recently, several parental percentage allocation

strategies were tested using QP's in (Goddard 2009; Pryce *et al.* 2012; Schierenbeck *et al.* 2011).

Box 19: Preparing the wheat data for analysis, setting up the optimization function

```
> Wheat.Ysc<-Wheat.Y
> Wheat.Ysc[,2]<-(Wheat.Ysc[,2]-mean(Wheat.Ysc[,2]))/sd(Wheat.Ysc[,2])
> P<-cbind(Wheat.Ysc, Wheat.M)
> rownames(P)<-rownames(Wheat.M)
> impinbreed=.1
> STPGAUSERDEFFUNC<-function(Train,Test=NULL, P,
+                             lambda=NULL, C=NULL){
+   trsmallM<-P[rownames(P)%in%Train,-c(1:2)]
+   g<-matrix(P[rownames(P)%in%Train,2], ncol=1)
+   A<-A.mat(M=trsmallM+1)
+   return(-(1-impinbreed)*mean(g)+
+           impinbreed*mean(c(A[lower.tri(A, diag=TRUE)])))
+ }
```

By solving the QP in Equation (6) for varying values of ρ , or using the similar criteria in the mate selection approaches, we can trace out an efficient frontier curve, a smooth non-decreasing curve that gives the best possible trade-off of genetic variance against gain. This curve represents the set of optimal allocations and it is called the efficiency frontier (EF) curve in finance (Markowitz 1952) and breeding literature.

Many practical applications require additional constraints, and it is possible to extend these formulations to introduce additional constraints as positiveness, minimum-maximum for proportions, minimum-maximum for number of lines (cardinality constraints). It is not too difficult to use **STPGA** to solve a version of this problem.

Suppose the breeder wants to select a subset of the individuals to become parents of the next generation increasing gain while controlling for coancestry and inbreeding. Note that the breeder only wants to select a subset and therefore we can assume that the parental contributions will be the same for each of the selected individuals. The following code illustrates how to select 10 individuals from the wheat data set for changing values of λ .

Box 20: Selected individuals for changing values of λ

```

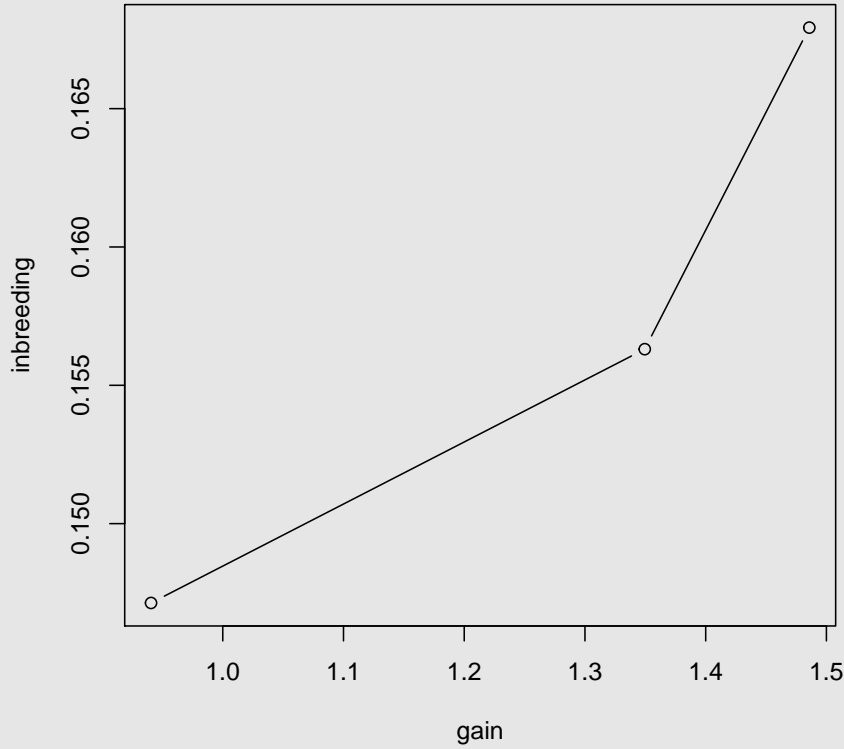
> GAOUTLIST<-vector(mode="list", length=5)
> i=1
> for (impinbreed in c(0.01,.95,.99)){
+   STPGAUSERDEFFUNC<-function(Train,Test=NULL, P, lambda=NULL, C=NULL){
+     trsmallM<-P[rownames(P)%in%Train,-c(1:2)]
+     g<-P[rownames(P)%in%Train,2]
+     A<-A.mat(M=trsmallM+1)
+     return(-(1-impinbreed)*mean(g)+
+             impinbreed*mean(c(A[lower.tri(A, diag=TRUE)])))
+   }
+   GAOUT<-GenAlgForSubsetSelectionNoTest(P=P, ntoselect=10,npop=100,
+     nelite=10, mutprob=.5, mutintensity = 1,niterations=100,
+     minitbefstop=50, tabumemsize = 1, plotiters=FALSE,tabu=FALSE,
+     lambda=1e-9,errorstat="STPGAUSERDEFFUNC", mc.cores=4)
+   GAOUTLIST[[i]]<-GAOUT
+   i=i+1
+ }
> GAMINvec<-c()
> GAMINvecgain<-c()
> GAMINvecinbreeding<-c()
> for (i in 1:3){
+   Train<-GAOUTLIST[[i]]$`Solution with rank 1`
+   trsmallM<-P[rownames(P)%in%Train,-c(1:2)]
+   g<-P[rownames(P)%in%Train,2]
+   A<-A.mat(M=trsmallM+1)
+   GAMINvec<-c(GAMINvec,
+     min(GAOUTLIST[[i]]$`Best criterion values over iterarions`))
+   GAMINvecgain<-c(GAMINvecgain, mean(g))
+   GAMINvecinbreeding<-c(GAMINvecinbreeding,mean(c(A[lower.tri(A, diag=TRUE)])))
+ }

```

After that I extract the average gain and inbreeding values for each value of λ and trace the frontier curve.

Box 21: Points on the frontier surface

```
> plot(GAMINvecgain, GAMINvecinbreeding, type="b",
+      xlab="gain", ylab="inbreeding")
```



Suppose now the breeder wants us to pick 10 individuals, but also asks for the best parental contribution proportions. This is a mixed integer quadratic programming problem. Let ϵ_i be the minimum proportion that must be allocated to line i , δ_i be the maximum proportion that must be allocated to line i if any of line i will be conserved, where we must have $0 \leq \epsilon_i \leq \delta_i \leq 1$. Introduce the binary variables:

$$z_i = \begin{cases} 1 & \text{if any of line } i \text{ is included,} \\ 0 & \text{otherwise.} \end{cases}$$

The cardinality constrained optimization problem is given by

$$\begin{aligned}
& \text{minimize} && r = c' \frac{A}{2} c \\
& \text{subject to} && c' b = \rho, \\
& && c' \mathbf{1} = 1, \\
& && \mathbf{c} \geq 0, \\
& && \sum_{i=1}^m z_i = K, \\
& && \epsilon_i z_i \leq c_i \leq \delta_i z_i, \quad i = 1, 2, \dots, m, \\
& && z_i \in \{0, 1\}, \quad i = 1, 2, \dots, m.
\end{aligned}$$

Here is the code for doing the same thing we have done with the previous example, except we provide parental contributions.

Box 22: Estimating the parental contributions for the cardinality constrained problem for changing values of λ

```

> require(quadprog)
> GAOUTLIST<-vector(mode="list", length=5)
> i=1
> for (impinbreed in c(0.01,.95,.99)){
+   STPGAUSERDEFFUNC<-function(Train,Test=NULL, P, lambda=1e-5, C=NULL){
+     trsmallM<-P[rownames(P)%in%Train,-c(1:2)]
+     g<-c(P[rownames(P)%in%Train,2])
+     A<-A.mat(M=trsmallM+1)
+     n=length(g)
+     sol <- solve.QP(Dmat=(n^2/2)*impinbreed*(A+lambda*diag(n)),
+       dvec=(1/n)*(1-impinbreed)*g, Amat=cbind(rep(1,n),diag(n),-diag(n)),
+       bvec=rbind(1,matrix(0,ncol=1,nrow=n),
+         matrix(-1,ncol=1,nrow=n)), meq=1)
+     names(sol$solution)<-rownames(trsmallM)
+     return(sol$value)
+   }
+   GAOUT<-GenAlgForSubsetSelectionNoTest(P=P, ntoselect=10,npop=50,
+     nelite=5, mutprob=.5, mutintensity = 1,niterations=100,
+     minitbefstop=50, tabumemsize = 1,plotiters=FALSE,tabu=FALSE,
+     lambda=1e-5,errorstat="STPGAUSERDEFFUNC", mc.cores=4)
+   GAOUTLIST[[i]]<-GAOUT
+   i=i+1
+ }

```

Box 23: Estimating the parental contributions for the cardinality constrained problem for changing values of λ

```
> GAMINvec<-c()
> GASols<-vector(mode="list")
> for (i in 1:3){
+   trsmallM<-P[rownames(P)%in%GAOUTLIST[[i]]$`Solution with rank 1`,~c(1:2)]
+   g<-c(P[rownames(P)%in%GAOUTLIST[[i]]$`Solution with rank 1`,2])
+   A<-A.mat(M=trsmallM+1)
+   n=length(g)
+   impinbreed<-c(0.01,.95,.99)[i]
+
+   sols<-solve.QP(Dmat=(n^2/2)*impinbreed*(A+1e-9*diag(n)),
+     dvec=(1/n)*(1-impinbreed)*g,Amat=cbind(rep(1,n),diag(n),-diag(n)),
+     bvec=rbind(1,matrix(0,ncol=1,nrow=n),
+       matrix(-1,ncol=1,nrow=n)), meq=1)$solution
+   names(sols)<-GAOUTLIST[[i]]$`Solution with rank 1`
+   GASols[[i]] <-sols
+   GAMINvec<-c(GAMINvec,
+     min(GAOUTLIST[[i]]$`Best criterion values over iterarions`))
+ }
> print(round(GASols[[2]], digits=3))
```

IWA8606856	PPG-1	AGRISS	IWA8606779	466A	H86-708	PI94479
0.1	0.1	0.1	0.1	0.1	0.1	0.1
ZG4163/73	NORIN10	PI254037				
0.1	0.1	0.1				

Variable selection in regression using model selection criteria

One of the standard use of GA is in variable selection and STPGA can be used in variable selection.

Another very popular method for variable selection and penalized (shrunk) parameter estimation in high dimensional regression is the lasso approach which can be summarized as finding the best regression coefficients that minimize the regression loss function while minimizing a multiple of the ℓ_1 norm of the coefficients. The relative stress on the importance of either of these complementary functions is expressed as a linear combination of these two. For example with the squared error loss the lasso optimization criterion is

$$(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda|\boldsymbol{\beta}|'\mathbf{1}.$$

The idea, its implications for many different fields of science can not be overlooked, the same goes for the enormous number of methods developed to solve this problem and its variations. However, there is a major statistical problem: there isn't enough principle behind its objective criterion so it is no big magic. It does not have certain properties we would like from a good regression. For example, the residuals of a model using the coefficients obtained by lasso are not orthogonal to the design of the variables that are selected by lasso. This last issue can

be circumvented by using lasso only as a selection operator and then fitting the coefficients with least squares without answering the main question: Why this criterion, not another one? Parsimony, yes, why this one? Is this the one that gives the best generalization error (minimize the prediction error)? The only good argument for the defense of this criterion I can think of is a Bayesian one and the so called "oracle" property which is only true under restrictive assumptions such as no dependency among the explanatory variables. we should be careful not to confuse the actual problem with the method, such as the carpenter who sees everything as nails since he/she is good at using hammers.

In my opinion, the whole area of norm penalized estimation neglects the many model selection criteria (AIC ([Akaike 1974](#)), BIC ([Schwarz et al. 1978](#)), ICOMP(IFIM) ([Bozdogan 1987](#)), etc,...) introduced during the 1970's and onward by many prominent statisticians whose derivations depends on the solid theory of likelihoods, divergence measures, etc.,.... I am not sure why we should develop elegant theories and then ignore them for not so elegant ones. So while lasso [Tibshirani \(1996\)](#) is great and can be solved at great speed in serial (after investment a huge amounts of energy and resources in the last 10 or more years) and the methods to find its solutions can only be partially parallelized. I am not sure the shrinkage approach can take on the next challenge of solving extremely large and complex problems. In addition to not being able to adopt well to parallel computer architectures, their application areas are limited by the problems these methods can address and addressing new problems with the same techniques involves inventing newly crafted methodologies which consumes the most amount of time and resources.

Here is an example using STPGA. I am going to demonstrate this with a classic body fat data set. The data is available in the package **UsingR** ([Verzani 2015](#)). I will use the AIC criterion, find the best subsets of size two through 12 (there are 13 explanatory variables) and obtain their AIC values.

Box 24: Variable selection for regression

```

> data("fat", package = "UsingR")
> mod <- lm(body.fat.siri ~ age + weight + height + neck + chest + abdomen +
+ hip + thigh + knee + ankle + bicep + forearm + wrist, data = fat)
> x <- model.matrix(mod)
> y <- model.response(model.frame(mod))
> fitnessfuncforSTPGA <- function(Train,Test=NULL, P, lambda=1e-6, C=NULL) {
+   X <- t(P[rownames(P)%in%Train,])
+   mod <- lm.fit(X, y)
+   class(mod) <- "lm"
+   return(AIC(mod))
+ }
> stpgaoutlist<-vector(mode="list")
> ii=1
> for (i in 2:(ncol(x)-2)){
+   stpgaoutlist[[ii]]<-GenAlgForSubsetSelectionNoTest(P=t(x[, -1]),
+     ntoselect=i, npop=200,
+     nelite=10, mutprob=.5, mutintensity = 1,
+     niterations=200, minitbefstop=50, tabu=FALSE,
+     tabumemsize = 1, plotiters=FALSE, lambda=1e-9,
+     errorstat="fitnessfuncforSTPGA", mc.cores=4)
+   ii=ii+1
+ }

```


Box 25: Extracting results

```

> GAMINs<-c()
> ii=1
> for (i in 2:(ncol(x)-2)){
+   GAMINs<-c(GAMINs,
+             min(stpgaoutlist[[ii]]$`Best criterion values over iterarions`))
+   ii=ii+1
+ }
> selectedvars6<-stpgaoutlist[[6]]$`Solution with rank 1`
> min(stpgaoutlist[[6]]$`Best criterion values over iterarions`)

[1] 1460.21

> selectedvars6

[1] "abdomen" "forearm" "hip"      "age"      "wrist"    "thigh"    "neck"

> selectedvars7<-stpgaoutlist[[7]]$`Solution with rank 1`
> min(stpgaoutlist[[7]]$`Best criterion values over iterarions`)

[1] 1459.716

> selectedvars7

[1] "forearm" "thigh"    "age"      "abdomen" "hip"      "wrist"    "height"
[8] "neck"

> selectedvars8<-stpgaoutlist[[8]]$`Solution with rank 1`
> min(stpgaoutlist[[8]]$`Best criterion values over iterarions`)

[1] 1459.474

> selectedvars8

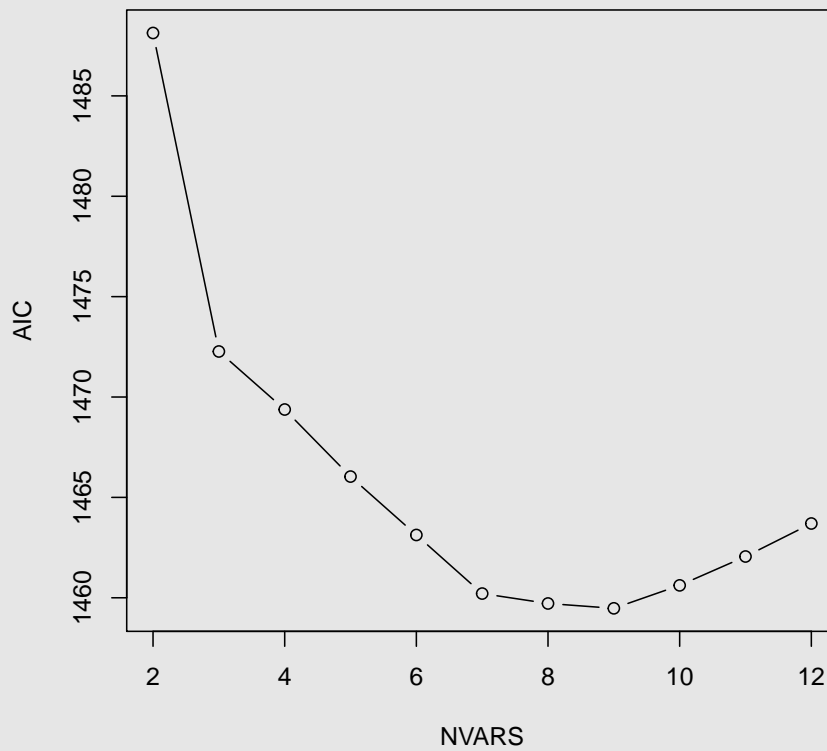
[1] "thigh"    "abdomen" "weight"   "neck"     "hip"      "forearm"  "age"
[8] "wrist"    "height"

```

Lets plot the results. It looks like we should pick seven or eight variables.

Box 26: Plotting AIC results for different subset sizes

```
> plot(2:(ncol(x)-2),GAMINs, type="b", xlab="NVARs", ylab="AIC")
```



Now, I will try to using an unsupervised approach to variable selection, the methodology is new. Why not pick the subset of variables that result in a kernel matrix that is most aligned with the the kernel that is calculated based on all 13 explanatory variables. Interestingly, most of the variables that are selected by supervised selection are also identified here.

Box 27: Variable selection- Unsupervised using kernel alignment

```

> Afull<-tcrossprod(scale(x[,-1], center=TRUE, scale=TRUE))/(ncol(x)-1)
> STPGAUSERDEFFUNC<-function(Train,Test=NULL, P, lambda=1e-6, C=NULL){
+   trsmallx<-t(scale(P[rownames(P)%in%Train,],center=TRUE, scale=TRUE))
+   Atr<-tcrossprod(trsmallx)/(ncol(trsmallx))
+   return(mean((c(Afull[lower.tri(Afull, diag=TRUE)]))
+               -c(Atr[lower.tri(Atr, diag=TRUE)]))^2))
+ }
> GAOUT<-GenAlgForSubsetSelectionNoTest(P=t(x[,-1]),
+   ntoselect=7,npop=200,
+   nelite=5, mutprob=.5, mutintensity = 1,
+   niterations=200, minitbefstop=50,tabu=FALSE,
+   tabumemsize = 1,plotiters=FALSE,lambda=1e-9,
+   errorstat="STPGAUSERDEFFUNC",mc.cores=4)
> selectedunsupervised<-GAOUT$`Solution with rank 1`
> selectedunsupervised

[1] "knee"      "neck"      "ankle"     "wrist"     "age"       "forearm"   "bicep"

> selectedvars6

[1] "abdomen" "forearm" "hip"      "age"       "wrist"     "thigh"     "neck"

> intersect(selectedvars6,selectedunsupervised)

[1] "forearm" "age"      "wrist"    "neck"

```

Here is a similar approach, we only add the response variable to the calculation of full data kernel.

Box 28: Variable selection - supervised using kernel alignment

```

> newx<-cbind(x[,-1],y)
> colnames(newx)

[1] "age"      "weight"   "height"   "neck"     "chest"    "abdomen"  "hip"
[8] "thigh"    "knee"     "ankle"    "bicep"    "forearm"  "wrist"    "y"

> Afull2<-tcrossprod(scale(newx, center=TRUE, scale=TRUE))/(ncol(x))
> STPGAUSERDEFFUNC2<-function(Train,Test=NULL, P, lambda=1e-6, C=NULL){
+   trsmallx<-t(scale(P[rownames(P)%in%Train,],center=TRUE, scale=TRUE))
+   Atr<-tcrossprod(trsmallx)/(ncol(trsmallx))
+
+   return(mean((c(Afull2[lower.tri(Afull2, diag=TRUE)]))
+                 -c(Atr[lower.tri(Atr, diag=TRUE)]))^2))
+ }
> GAOUT2<-GenAlgForSubsetSelectionNoTest(P=t(x[,-1]),
+   ntoselect=7,npop=200,
+   nelite=5, mutprob=.5, mutintensity = 1,
+   niterations=200, minitbefstop=50,tabu=FALSE,
+   tabumemsize = 1,plotiters=FALSE,lambda=1e-9,
+   errorstat="STPGAUSERDEFFUNC2",mc.cores=4)
> selectedsupervised<-GAOUT2$`Solution with rank 1`
> selectedsupervised

[1] "forearm" "knee"    "ankle"    "wrist"    "age"      "neck"     "bicep"

> selectedvars6

[1] "abdomen" "forearm" "hip"      "age"      "wrist"    "thigh"    "neck"

> intersect(selectedvars6,selectedunsupervised)

[1] "forearm" "age"      "wrist"    "neck"

> intersect(selectedvars6,selectedsupervised)

[1] "forearm" "age"      "wrist"    "neck"

> intersect(selectedunsupervised,selectedsupervised)

[1] "knee"    "neck"    "ankle"    "wrist"    "age"      "forearm" "bicep"

```

We can also align to the kernel matrix obtained from only the response variable(s):

Box 29: Variable selection- supervised (only by response) using kernel alignment

```

> Afull3<-tcrossprod(scale(y, center=TRUE, scale=TRUE))
> STPGAUSERDEFFUNC3<-function(Train,Test=NULL, P, lambda=1e-6, C=NULL){
+   trsmallx<-t(scale(P[rownames(P)%in%Train,],center=TRUE, scale=TRUE))
+   Atr<-tcrossprod(trsmallx)/(ncol(trsmallx))
+
+   return(mean((c(Afull3[lower.tri(Afull3, diag=TRUE)]))
+               -c(Atr[lower.tri(Atr, diag=TRUE)]))^2))
+ }
> GAOUT3<-GenAlgForSubsetSelectionNoTest(P=t(x[, -1]),
+   ntoselect=7, npop=200,
+   nelite=5, mutprob=.5, mutintensity = 1,
+   niterations=200, minitbefstop=50, tabu=FALSE,
+   tabumemsize = 1, plotiters=FALSE, lambda=1e-9,
+   errorstat="STPGAUSERDEFFUNC3", mc.cores=4)
> selectedsupervised2<-GAOUT3$`Solution with rank 1`
> selectedsupervised2

[1] "wrist"   "bicep"   "neck"    "forearm" "ankle"   "knee"    "age"

> selectedvars6

[1] "abdomen" "forearm" "hip"      "age"      "wrist"    "thigh"    "neck"

> intersect(selectedvars6,selectedunsupervised)

[1] "forearm" "age"      "wrist"    "neck"

> intersect(selectedvars6,selectedsupervised2)

[1] "forearm" "age"      "wrist"    "neck"

> intersect(selectedunsupervised,selectedsupervised2)

[1] "knee"    "neck"    "ankle"   "wrist"   "age"     "forearm" "bicep"

> intersect(selectedsupervised,selectedsupervised2)

[1] "forearm" "knee"    "ankle"   "wrist"   "age"     "neck"    "bicep"

```

Let's see which variables are picked by lasso and best subsets regressions:

Box 30: Variable selection- supervised using lasso and best subsets regression

```

> # lasso
> library(glmnet)
> fit.glmnet.lasso.cv <- cv.glmnet(as.matrix(scale(x[,-1], center=T, scale=T)),
+                               as.matrix(y, ncol=1),
+                               nfold = 5,
+                               alpha = 1)
> coef(fit.glmnet.lasso.cv, s = fit.glmnet.lasso.cv$lambda.1se)

14 x 1 sparse Matrix of class "dgCMatrix"
              1
(Intercept) 19.1507937
age          0.2887603
weight       .
height      -0.5184681
neck         .
chest        .
abdomen      6.2447837
hip          .
thigh        .
knee         .
ankle        .
bicep        .
forearm      .
wrist       -0.1999128

> GAOUT4<-GenAlgForSubsetSelectionNoTest(P=t(x[,-1]),
+   ntoselect=4,npop=200,
+   nelite=5, mutprob=.5, mutintensity = 1,
+   niterations=200, minitbefstop=50,tabu=FALSE,
+   tabumemsize = 1,plotiters=FALSE,lambda=1e-9,
+   errorstat="STPGAUSERDEFFUNC3",mc.cores=4)
> selectedsupervised4<-GAOUT4$`Solution with rank 1`
> selectedsupervised4

[1] "knee"    "neck"    "forearm" "age"

```

Box 31: Variable selection- supervised using lasso and best subsets regression

```

> # lasso
> library(leaps)
> regsubsets.out <-
+   regsubsets(body.fat.siri ~ age + weight + height + neck + chest + abdomen +
+     hip + thigh + knee + ankle + bicep + forearm + wrist,
+     data = fat,
+     nbest = 1, nvmax = NULL,
+     force.in = NULL, force.out = NULL,
+     method = "exhaustive")
> sevenvarsselectedbyleaps<-colnames(as.data.frame(summary(regsubsets.out)$outmat))[
+   which(as.data.frame(summary(regsubsets.out)$outmat)[7,]=="*")]
> intersect(sevenvarsselectedbyleaps,selectedunsupervised)

[1] "age"      "neck"      "forearm" "wrist"

> intersect(sevenvarsselectedbyleaps,selectedvars7)

[1] "age"      "neck"      "abdomen" "thigh"     "forearm" "wrist"

```

Now, again, lets to something that hasn't been done before: I am going to "make up" a variable selection criterion for regression that has some nice properties. Lets assume as before \mathbf{y} is the length n response vector, X is the $n \times p$ design matrix; lets denote the generic design of $k \leq p$ variables as $X^{(k)}$ and the set of such k variable designs as $\mathbf{X}^{(k)}$. For this generic design matrix we want to find $\beta^{(k)} \in \mathbf{R}^k$ such that error sum of squares are minimized, given $X^{(k)}$ this is the least squares solution $\widehat{\beta}^{(k)} = (X^{(k)'}X^{(k)})^{-1}X^{(k)'}\mathbf{y}$ with covariance matrix proportional to $(X^{(k)'}X^{(k)})^{-1}$. The optimization problem is stated as

$$(\widehat{X}^{(k)}, \widehat{\beta}^{(k)}) = \underset{X^{(k)} \in \mathbf{X}^{(k)}, \beta^{(k)} \in \mathbf{R}^k}{\operatorname{argmin}} (\mathbf{y} - X^{(k)}\beta^{(k)})'(\mathbf{y} - X^{(k)}\beta^{(k)}) - \lambda |X^{(k)'}X^{(k)}|$$

and to simplify it further, noting that the penalty term does not depend on $\beta^{(k)}$, we can rewrite the above as

$$\widehat{X}^{(k)} = \underset{X^{(k)} \in \mathbf{X}^{(k)}}{\operatorname{argmin}} (\mathbf{y}(I - X^{(k)}(X^{(k)'}X^{(k)})^{-1}X^{(k)'}\mathbf{y} - \lambda \log |X^{(k)'}X^{(k)}|$$

and $\widehat{\beta}^{(k)} = (\widehat{X}^{(k)'}\widehat{X}^{(k)})^{-1}\widehat{X}^{(k)'}\mathbf{y}$. The interpretation of this optimization criterion is as follows, we want the set of k variables that minimize the squared error loss while also keeping the variances and correlations among the estimated coefficients as low as possible, by increasing λ we would be increasing the the relative importance given to the variance and covariances of the coefficients. Note that, when the information matrix is singular, we can use a generalized inverse obtained by regularization (i.e. add a small constant to the diagonal elements) as described in the previous sections.

Box 32: Variable selection

```

> mod <- lm(body.fat.siri ~ age + weight + height + neck + chest + abdomen +
+ hip + thigh + knee + ankle + bicep + forearm + wrist, data = fat)
> x <- scale(as.matrix(model.matrix(mod)), center=F, scale=T)
> y <- model.response(model.frame(mod))
> y<-y/sd(y)
> fitnessfuncforSTPGA <- function(Train,Test=NULL, P, lambda=.5, C=NULL) {
+   X <- t(P[rownames(P)%in%Train,])
+   n<-nrow(X)
+
+       p<-ncol(X)
+       mindim<-min(p,n)
+       rownames(X)<-NULL
+       svdX<-svd(X, nu=mindim,nv=mindim)
+       insvdnonzero<-1:mindim
+       diagvecforinv<-(svdX$d[insvdnonzero])/((svdX$d[insvdnonzero])^2+1e-7)
+       coef<-tcrossprod(svdX$v*%diag(diagvecforinv),svdX$v)*%t(X)*%(y-mean(y))
+       resids<-y-X*%coef-mean(y)
+       out<-(1-lambda)*mean(resids^2)-
+       lambda*determinant(crossprod(X), logarithm = T )$modulus
+   return(out)
+ }
> GAMINSmat<-c()
> lambda=1e-6
> stpgaoutlist<-vector(mode="list")
> ii=1
> for (i in 2:(ncol(x)-1)){
+   stpgaoutlist[[ii]]<-GenAlgForSubsetSelectionNoTest(P=t(x[, -1]),
+       ntoselect=i, npop=100,
+       nelite=10, mutprob=.5, mutintensity = 1,
+       niterations=100, minitbefstop=50, tabu=FALSE,
+       tabumemsize = 1, plotiters=F, lambda=lambda,
+       errorstat="fitnessfuncforSTPGA", mc.cores=4)
+   ii=ii+1
+ }
> GAMINs<-c()
> ii=1
> for (i in 2:(ncol(x)-1)){
+   GAMINs<-c(GAMINs,
+       min(stpgaoutlist[[ii]]$`Best criterion values over iterarions`))
+   ii=ii+1
+ }
> GAMINSmat<-cbind(GAMINSmat, GAMINs)
>

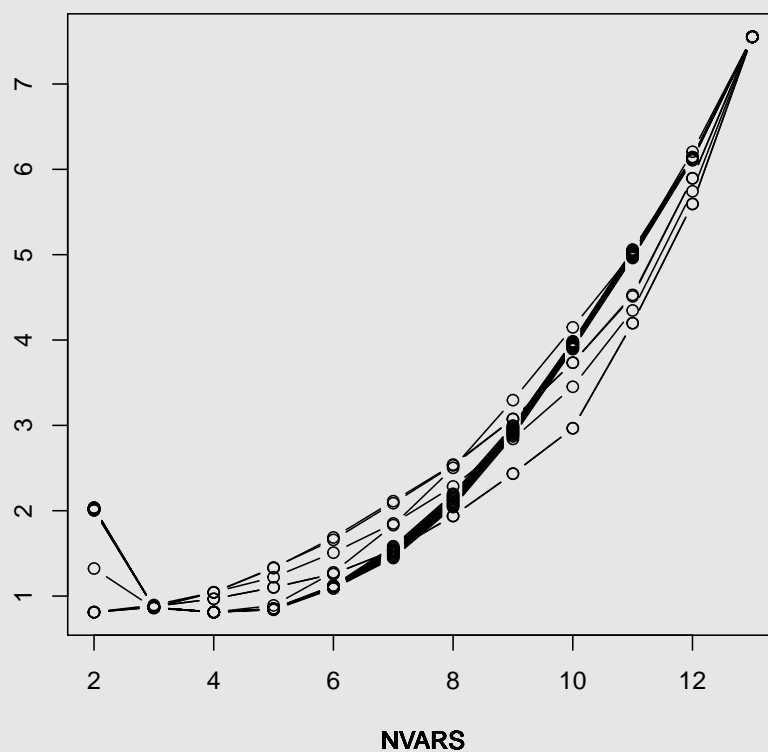
```


Box 33: Variable selection

```

> plot(2:(ncol(x)-1),GAMINs, type="b", xlab="NVARs", ylab="")
> for (lambda in c(seq(1e-5,.2, length=3),seq(0.21,.90, length=3),seq(0.91,1, length=10))) {
+   par(new=T)
+   stpgaoutlist<-vector(mode="list")
+   ii=1
+   for (i in 2:(ncol(x)-1)){
+     stpgaoutlist[[ii]]<-GenAlgForSubsetSelectionNoTest(P=t(x[, -1]),
+       ntoselect=i, npop=100,
+       nelite=10, mutprob=.5, mutintensity = 1,
+       niterations=100, minitbefstop=50, tabu=FALSE,
+       tabumemsize = 1, plotiters=FALSE, lambda=lambda,
+       errorstat="fitnessfuncforSTPGA", mc.cores=4)
+     ii=ii+1
+   }
+   GAMINs<-c()
+   ii=1
+   for (i in 2:(ncol(x)-1)){
+     GAMINs<-c(GAMINs,min(stpgaoutlist[[ii]]$`Best criterion values over iterarions`))
+     ii=ii+1
+   }
+   GAMINsmat<-cbind(GAMINsmat,GAMINs)
+   plot(2:(ncol(x)-1),GAMINs, type="b", xlab="NVARs", ylab="", axes=F)
+ }

```



Box 34: Variable selection- new criterion

```
> apply(GAMINsMat, 2, which.min)

GAMINs GAMINs GAMINs GAMINs GAMINs GAMINs GAMINs GAMINs GAMINs GAMINs GAMINs
      1      1      1      1      1      3      3      3      3      3      3
GAMINs GAMINs GAMINs GAMINs GAMINs GAMINs
      3      3      3      3      3      3

> apply(GAMINsMat, 2, min)

      GAMINs      GAMINs      GAMINs      GAMINs      GAMINs      GAMINs
0.8111831  0.8111216  0.1281011 -0.5696401 -0.6409756 -4.4104119
      GAMINs      GAMINs      GAMINs      GAMINs      GAMINs      GAMINs
-9.0394933 -9.1736695 -9.3078458 -9.4420221 -9.5761984 -9.7103746
      GAMINs      GAMINs      GAMINs      GAMINs      GAMINs
-9.8445509 -9.9787272 -10.1129034 -10.2470797 -10.3812560

> newselcritout<-GenAlgForSubsetSelectionNoTest(P=t(x[,-1]),
+       ntoselect=5,npop=200,
+       nelite=10, mutprob=.5, mutintensity = 1,
+       niterations=200, minitbefstop=50,tabu=FALSE,
+       tabumemsize = 1,plotiters=F,lambda=.1,
+       errorstat="fitnessfuncforSTPGA",mc.cores=4)
> newselcritout$`Solution with rank 1`

[1] "forearm" "ankle"  "height"  "chest"   "wrist"
```

Note that the design criterion above defines a class of optimal solutions whose criterion values that can be plotted as a 3-dimensional surface for changing λ and n ; this is similar to the frontier curve discussed above for balancing gains and inbreeding problem. In general, any multiobjective optimization problem defines a surface of Pareto optimal points that is called the frontier surface which may or may not be continuous. Frontier surfaces are inspected for identifying a good solution among and in the case of variable selection should be useful for identifying a single model from the class of optimal solutions.

It is easy to make criteria like the one I have made up. Here is another one

$$\widehat{X^{(k)}} = \underset{X^{(k)} \in \mathbf{X}^{(k)}}{\operatorname{argmin}} (\mathbf{y}(I - X^{(k)}(X^{(k)'}X^{(k)})^{-1}X^{(k)'})\mathbf{y} + \lambda \operatorname{trace}(X^{(k)'}X^{(k)})^{-1})$$

and $\widehat{\beta^{(k)}} = (\widehat{X^{(k)}}'\widehat{X^{(k)}})^{-1}\widehat{X^{(k)}}'\mathbf{y}$; and another one:

$$\widehat{X^{(k)}} = \underset{X^{(k)} \in \mathbf{X}^{(k)}}{\operatorname{argmin}} (\mathbf{y}(I - X^{(k)}(X^{(k)'}X^{(k)})^{-1}X^{(k)'})\mathbf{y} + \lambda \operatorname{trace}X^{(k)}(X^{(k)'}X^{(k)})^{-1}X^{(k)'})$$

and $\widehat{\beta^{(k)}} = (\widehat{X^{(k)}}'\widehat{X^{(k)}})^{-1}\widehat{X^{(k)}}'\mathbf{y}$. Both of these have nice interpretation as well. In fact, any

optimal design criteria of Section 4.1 could be relevant in this context, for example, try

$$\widehat{X^{(k)}} = \underset{X^{(k)} \in \mathbf{X}^{(k)}}{\operatorname{argmin}} (\mathbf{y}(I - X^{(k)}(X^{(k)'}X^{(k)})^{-1}X^{(k)'})\mathbf{y} + \lambda \operatorname{trace} X^{*(k)}(X^{(k)'}X^{(k)})^{-1}X^{*(k)'}),$$

and $\widehat{\beta^{(k)}} = (\widehat{X^{(k)}}'\widehat{X^{(k)}})^{-1}\widehat{X^{(k)}}'\mathbf{y}$. I encourage the user to program these criterion or their own, perhaps the one you invent will make more sense for the problem that you are trying to solve than this or that other criterion.

The main point of these examples is that variable selection problem can be viewed as a multiobjective optimization problem, where a loss function and a penalty function are being simultaneously optimized. The loss function measures the fit of data to the training data and its optimal value for the optimal fixed sized set of variables improves as more variables are added to that model. The penalty function measures the overall quality of the model. Usually deteriorates as more variables are added to a model. In general, we can assume that the loss function and the penalty function are conflicting objectives, i.e., there does not exist a single solution that simultaneously optimizes both functions. This leads to a possibly infinite number of Pareto optimal solutions. The set of non-dominated solutions (none of the objective functions can be improved in value without degrading some other) are called Pareto optimal, and define a surface called the Pareto frontier.

Identifying influential observations in regression (and keeping the most consistent regression data)

When a regression model is fitted to data, if a few of the observations are different in some way from the bulk of the data then using all observations into the model might not be appropriate. The inclusion or exclusion of a few of these observations make a significant change in the parameter estimates or predictions. These are referred to as influential observations. One statistic that is used to identify the influence of an observation is the so called DFBETAS which measures the effect of deletion of single observations on the estimated model coefficients. This measure can be generalized to deletion of a group of individuals as

$$DFBETAS(X_{(-)}) = \frac{1}{\widehat{\sigma_{(-)}^2}} (\widehat{\beta} - \widehat{\beta_{(-)}})' (X_{(-)}'X_{(-)})^{-1} (\widehat{\beta} - \widehat{\beta_{(-)}}),$$

where $\widehat{\beta}$ is the estimated regression coefficients from full data, $\widehat{\beta_{(-)}}$ is the same estimated using a part of the data (leaving out its complement from the analysis), $\widehat{\sigma_{(-)}^2}$ is the residual variance estimate obtained using only partial data and $X_{(-)}$ is the design matrix for partial data. Here is the application of this to the "iris" data set (available with base R) to locate the influential observations if there are any:

Box 35: Deleting influential obs in regression

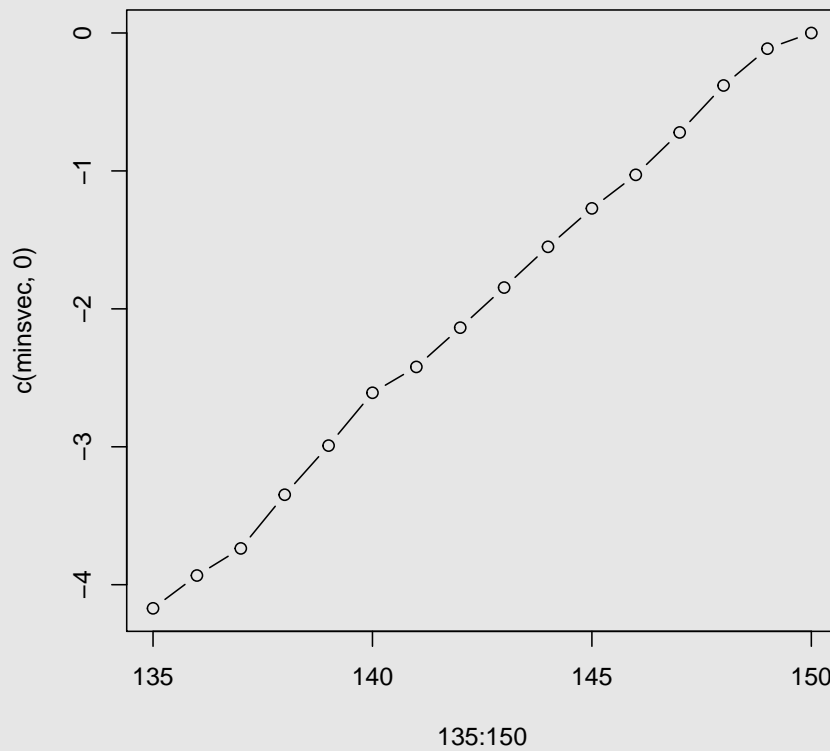
```

> library(STPGA)
> STPGAUSERFUNCI<-function(Train,Test=NULL, P, lambda=1e-6, C=NULL){
+   PTrain<-P[rownames(P)%in%Train,]
+   PtP<-crossprod(cbind(1,P[, -1]))
+   PTtPT<-crossprod(cbind(1,PTrain[, -1]))
+   B<-solve(PtP+lambda*diag(nrow(PtP)),crossprod(cbind(1,P[, -1]),P[, 1]))
+   BT<-solve(PTtPT+lambda*diag(nrow(PtP)),
+             crossprod(cbind(1,PTrain[, -1]),PTrain[, 1]))
+   resT<-PTrain[, 1]-cbind(1,PTrain[, -1])%*%BT
+   meanD<-mean((1/sd(resT))*diag(crossprod((B-BT),
+                                             (PTtPT+lambda*diag(nrow(PtP)))*%*(B-BT))))
+   return(-meanD)
+ }
> data(iris)
> P<-as.matrix(scale(iris[,1:4], center=TRUE, scale=TRUE))
> rownames(P)<-rownames(iris)
> STPGAoutlist<-vector(mode="list", length=20)
> ii=1
> for (i in 135:149){
+   STPGAoutlist[[ii]]<-GenAlgForSubsetSelectionNoTest(P=P,ntoselect=i,
+             mutprob = .8, npop = 100, nelite = 5, keepbest = TRUE,
+             tabu = F, tabumemsize = 0, mutintensity = 1,plotiters=FALSE,
+             niterations=200, minitbefstop = 50,
+             errorstat = "STPGAUSERFUNCI", mc.cores=4)
+   ii=ii+1
+ }
> minsvec<-c()
> ii=1
> for (i in 135:149){
+   minsvec<-c(minsvec,
+             min(STPGAoutlist[[ii]]$`Best criterion values over iterarions`))
+   ii=ii+1
+ }

```

Box 36: Deleting influential obs in regression

```
> plot(135:150, c(minsvec, 0), type="b")
```



What we are looking in the above graph is a point of sharp increase. I don't see such clear change point, so let's replace replace six observations' values so that they are different than the rest:

Box 37: Deleting influential observations in regression

```

> data(iris)
> P<-as.matrix(scale(iris[,1:4], center=TRUE, scale=TRUE))
> rownames(P)<-rownames(iris)
> P[c(5,30,55,80,105,130),c(1:4)]<-
+   1.5*P[c(5,30,55,80,105,130),c(1:4)]*%*matrix(rnorm(16),nrow=4, ncol=4)
> STPGAoutlist<-vector(mode="list", length=20)
> ii=1
> for (i in 135:149){
+   STPGAoutlist[[ii]]<-GenAlgForSubsetSelectionNoTest(P=P,ntoselect=i,
+     mutprob = .8, npop = 100, nelite = 5, keepbest = TRUE,
+     tabu = F, tabumemsize = 0, mutintensity = 1,plotiters=FALSE,
+     niterations=200, minitbefstop = 50,
+     errorstat = "STPGAUSERFUNCI", mc.cores=4)
+   ii=ii+1
+ }
> minsvec<-c()
> ii=1
> for (i in 135:149){
+   minsvec<-c(minsvec,
+     min(STPGAoutlist[[ii]]$`Best criterion values over iterarions`))
+   ii=ii+1
+ }

```

The following plot shows that the criterion value shows a sharp increase when we go from 144 to 145 observations and there are total of 150 observations in the data set, six observations that aren't included in the set of 144 observations cause large change in the value of the estimated coefficients. We observe that these are the observations we have manipulated.

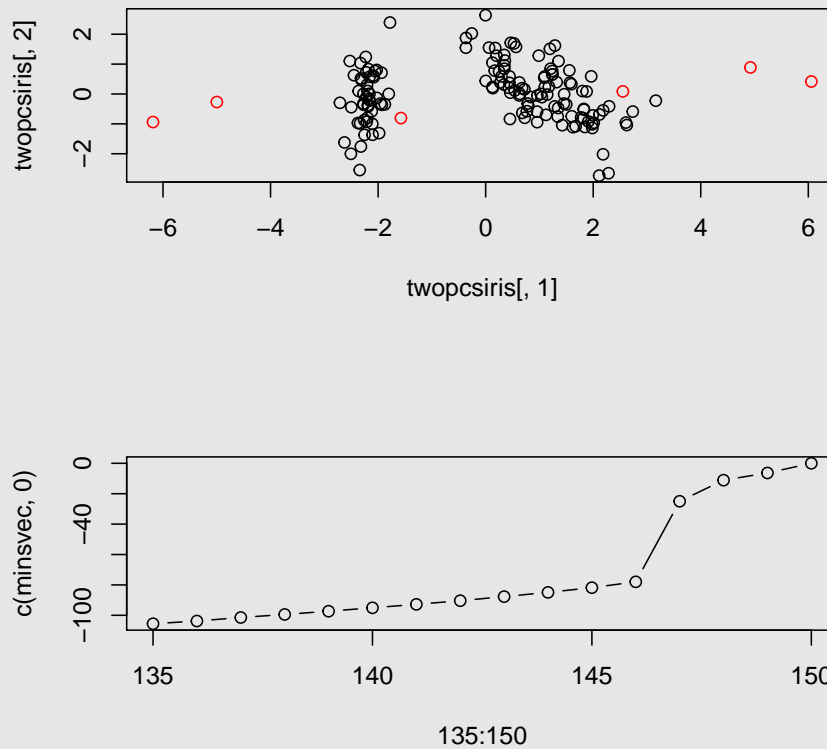
Box 38: Deleting influential observations in regression

```

> par(mfrow=c(2,1))
> twopcsiris<-(P%*%svd(P, nu=0,nv=2)$v)
> plot(twopcsiris[,1],twopcsiris[,2], col=(1:150)%in%c(5,30,55,80,105,130)+1)
> plot(135:150,c(minsvec,0), type="b")
> par(mfrow=c(1,1))
> STPGAout<-GenAlgForSubsetSelectionNoTest(P=P,ntoselect=144,
+           mutprob = .8, npop = 50, nelite = 5, keepbest = TRUE,
+           tabu = F, tabumemsize = 0, mutintensity = 1,plotiters=FALSE,
+           niterations=200, minitbefstop=50,
+           errorstat = "STPGAUSERFUNCI", mc.cores=4)
> intersect(STPGAout$`Solution with rank 1`, as.character(c(5,30,55,80,105,130)))

[1] "55" "5"

```



5. Concluding Remarks

I hope that I have provided enough examples to show that generic optimal subset selection problem is a useful tool. The list of examples and applications can easily be extended.

However, I think this should take some time to digest.

In STPGA package I have implemented a GA algorithm for subset selection which was inspired by the recent developments in the genomic breeding of plants, animals and other organisms. The main advantage of this algorithm is that it can be run in parallel to solve the subset selection problem for any given objective function.

I also did not compare the relative speed of this particular implementation of LA-GA-T algorithm to any other software. I admit that this software could be written so that it functioned faster using one processor and, no matter what I do, it will never work as fast as some other algorithms that are specialized in their tasks. However, I note that LA-GA-T algorithm can be run in parallel on many computers to solve problems as fast as some problem specific algorithms that have to be performed in serial.

Prediction accuracy from genomic models can be improved by targeting more informative individuals in the data set used to generate the predictors, this result has been exemplified by several papers in the area ([Akdemir *et al.* \(2015\)](#); [Yu *et al.* \(2016\)](#)). Nevertheless, the subject deserves more attention.

Some of the criteria I have mentioned in the selection of training populations section of this paper are not among the default criteria listed in Table 1, however, I have included them as user defined criteria in the help files that is provided with the package.

I have plans to migrate all of this program to a more efficient programming language for performance improvements, and there is room for improvement redesigning parts of the algorithm for However, there are some advantages of using R. Some of these include like the accessibility, availability, being able to define functions on the fly. Therefore, I will still be supporting this pure R version. As a statistician, I have the feeling that writing efficient software is partially out of my expertise, immediate interests and surely can be done much easier with collaboration with people with the right skills. As of now, I can use STPGA with moderately complex problems, either by allowing the algorithm to take its time or by using a highly parallelizable machine. I am open to any suggestions, collaborations in this respect.

Acknowledgements

I am grateful to family, Mehmet Ali and Güler and Ümit Özgür, Pelin May , Kristy Akdemir.

My work was also supported by tax payers through the grants: R01GM099992, R01GM101219 and USDA-NIFA-AFRI Triticeae Coordinated Agricultural Project- award number 2011-68002-30029.

Author contributions

Every idea, code in this manuscript was conceived by Deniz Akdemir unless they were acknowledged by the references within the article. If a person or a group makes any claims that they have contributed to this work in any way it should not be taken seriously.

References

- Akaike H (1974). “A new look at the statistical model identification.” *IEEE transactions on automatic control*, **19**(6), 716–723.
- Akdemir D, Godfrey OU (2015). *EMMREML: Fitting Mixed Models with Known Covariance Structures*. R package version 3.1, URL <https://CRAN.R-project.org/package=EMMREML>.
- Akdemir D, Jannink JL (2015). “Locally epistatic genomic relationship matrices for genomic association and prediction.” *Genetics*, **199**(3), 857–871.
- Akdemir D, Sánchez JI (2016). “Efficient Breeding by Genomic Mating.” *Frontiers in Genetics*, **7**.
- Akdemir D, Sanchez JI, Jannink JL (2015). “Optimization of genomic selection training populations with a genetic algorithm.” *Genet Sel Evol*, **47**, 38.
- Amdahl GM (1967). “Validity of the single processor approach to achieving large scale computing capabilities.” In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pp. 483–485. ACM.
- Ardia D, Mullen K, Peterson B, Ulrich J, Boudt K, Mullen MK (2016). “Package ‘DEoptim’.”
- Atkinson A, Donev A (1992). “Optimum experimental designs. Clarendon.”
- Box GE, Hunter WG, Hunter JS (1978). *Statistics for experimenters: an introduction to design, data analysis, and model building*, volume 1. JSTOR.
- Bozdogan H (1987). “ICOMP: A new model-selection criterion.” In *1. Conference of the International Federation of Classification Societies*, pp. 599–608.
- Brisbane J, Gibson J (1995). “Balancing selection response and rate of inbreeding by including genetic relationships in selection decisions.” *Theoretical and Applied Genetics*, **91**(3), 421–431.
- Burton PR, Clayton DG, Cardon LR, Craddock N, Deloukas P, Duncanson A, Kwiatkowski DP, McCarthy MI, Ouwehand WH, Samani NJ, *et al.* (2007). “Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls.” *Nature*, **447**(7145), 661–678.

- Chun H, Keleş S (2010). “Sparse partial least squares regression for simultaneous dimension reduction and variable selection.” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **72**(1), 3–25.
- Cressie N (1988). “Spatial prediction and ordinary kriging.” *Mathematical Geology*, **20**(4), 405–421. ISSN 0882-8121. doi:10.1007/BF00892986. URL <http://dx.doi.org/10.1007/BF00892986>.
- Crossa J, de los Campos G, Pérez P, Gianola D, Burgueno J, Araus JL, Makumbi D, Singh RP, Dreisigacker S, Yan JB, Arief V, Banziger M, Braun HJ (2010). “Prediction of Genetic Values of Quantitative Traits in Plant Breeding Using Pedigree and Molecular Markers.” *Genetics*, **186**(2), 713–U406.
- Crossa J, Jarquín D, Franco J, Pérez-Rodríguez P, Burgueño J, Saint-Pierre C, Vikram P, Sansaloni C, Petrolis C, Akdemir D, *et al.* (2016). “Genomic prediction of gene bank wheat landraces.” *G3: Genes| Genomes| Genetics*, **6**(7), 1819–1834.
- Daetwyler HD, Calus MP, Pong-Wong R, de los Campos G, Hickey JM (2013). “Genomic prediction in animals and plants: simulation of data, validation, reporting, and benchmarking.” *Genetics*, **193**(2), 347–365.
- Draper NR, Pukelsheim F (1996). “An overview of design of experiments.” *Statistical Papers*, **37**(1), 1–32.
- Fedorov VV (1972). *Theory of optimal experiments*. Elsevier.
- Fisher RA (1960). “The design of experiments. ed.” *New York: Hafner*.
- Fisher RA (1992). “The arrangement of field experiments.” In *Breakthroughs in statistics*, pp. 82–91. Springer.
- Furnival GM, Wilson RW (1974). “Regressions by leaps and bounds.” *Technometrics*, **16**(4), 499–511.
- Gianola D, Fernando RL, Stella A (2006). “Genomic-assisted Prediction of Genetic Value with Semiparametric Procedures.” *Genetics*, **173**(3), 1761–1776.
- Goddard M (2009). “Genomic selection: prediction of accuracy and maximisation of long term response.” *Genetics*, **136**(2), 245–257.
- Goldberg DE, Holland JH (1988). “Genetic algorithms and machine learning.” *Machine learning*, **3**(2), 95–99.

- González-Recio O, Gianola D, Long N, Weigel KA, Rosa GJM, Avendano S (2008). “Nonparametric Methods for Incorporating Genomic Information Into Genetic Evaluations: An Application to Mortality in Broilers.” *Genetics*, **178**(4), 2305–2313. doi:10.1534/genetics.107.084293. <http://www.genetics.org/content/178/4/2305.full.pdf+html>, URL <http://www.genetics.org/content/178/4/2305.abstract>.
- Gruska J (1999). *Quantum computing*, volume 2005. McGraw-Hill London.
- Haines LM (1987). “The application of the annealing algorithm to the construction of exact optimal designs for linear-regression models.” *Technometrics*, **29**(4), 439–447.
- Hayes B, Bowman P, Chamberlain A, Goddard M (2009). “Invited review: Genomic selection in dairy cattle: Progress and challenges.” *Journal of Dairy Science*, **92**(2), 433 – 443. ISSN 0022-0302.
- Henderson CR (1953). “Estimation of variance and covariance components.” *Biometrics*, **9**(2), 226–252.
- Henderson CR (1975). “Best linear Unbiased Estimation and Prediction Under a Selection Model.” *Biometrics*, **31**(2), 423–447.
- Henderson CR, Kempthorne O, Searle SR, Von Krosigk C (1959). “The estimation of environmental and genetic trends from records subject to culling.” *Biometrics*, **15**(2), 192–218.
- Holland JH (1992a). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press.
- Holland JH (1992b). “Genetic algorithms.” *Scientific american*, **267**(1), 66–72.
- Isidro J, Jannink JL, Akdemir D, Poland J, Heslot N, Sorrells ME (2015). “Training set optimization under population structure in genomic selection.” *Theoretical and Applied Genetics*, **128**(1), 145–158.
- Jannink JL (2010). “Dynamics of long-term genomic selection.” *Genetics Selection Evolution*, **42**(1), 35.
- Kempthorne O, *et al.* (1957). “An introduction to genetic statistics.” *An introduction to genetic statistics*.
- Kiefer J (1959). “Optimum experimental designs.” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 272–319.
- Kiefer JC, Brown L, Olkin I, Sacks J (1985). *Jack Carl Kiefer Collected Papers: Design of Experiments*. Springer.

- Laloë D (1993). “Precision and information in linear models of genetic evaluation.” *Genetics Selection Evolution*, **25**(6), 557–576.
- Laloë D, Phocas F (2003). “A proposal of criteria of robustness analysis in genetic evaluation.” *Livestock Production Science*, **80**(3), 241 – 256. ISSN 0301-6226. doi:[http://dx.doi.org/10.1016/S0301-6226\(02\)00092-1](http://dx.doi.org/10.1016/S0301-6226(02)00092-1). URL [//www.sciencedirect.com/science/article/pii/S0301622602000921](http://www.sciencedirect.com/science/article/pii/S0301622602000921).
- Leuenberger MN, Loss D (2001). “Quantum computing in molecular magnets.” *Nature*, **410**(6830), 789–793.
- Liu Q, Wang L, Frutos AG, Condon AE, Corn RM, Smith LM (2000). “DNA computing on surfaces.” *Nature*, **403**(6766), 175–179.
- Makowsky R, Pajewski NM, Klimentidis YC, Vazquez AI, Duarte CW, Allison DB, de Los Campos G (2011). “Beyond missing heritability: prediction of complex traits.” *PLoS Genet*, **7**(4), e1002051.
- Markowitz H (1952). “Portfolio selection.” *The journal of finance*, **7**(1), 77–91.
- Mebane Jr WR, Sekhon JS, Sekhon MJS (2015). “Package ‘rgenoud’.”
- Meuwissen T (1997). “Maximizing the response of selection with a predefined rate of inbreeding.” *Journal of animal science*, **75**(4), 934–940.
- Meuwissen THE, Hayes BJ, Goddard ME (2001). “Prediction of Total Genetic Value Using Genome-Wide Dense Marker Maps.” *Genetics*, **157**(4), 1819–1829.
- Mitchell TJ (1974). “An algorithm for the construction of “D-optimal” experimental designs.” *Technometrics*, **16**(2), 203–210.
- Nguyen NK, Miller AJ (1992). “A review of some exchange algorithms for constructing discrete D-optimal designs.” *Computational Statistics & Data Analysis*, **14**(4), 489–498.
- Paun G, Rozenberg G, Salomaa A (2005). *DNA computing: new computing paradigms*. Springer Science & Business Media.
- Pritchard DJ, Bacon DW (1978). “Prospects for reducing correlations among parameter estimates in kinetic models.” *Chemical Engineering Science*, **33**(11), 1539–1543.
- Pronzato L (2008). “Optimal experimental design and some related control problems.” *Automatica*, **44**(2), 303–325.
- Pronzato L, Müller WG (2012). “Design of computer experiments: space filling and beyond.” *Statistics and Computing*, **22**(3), 681–701.

- Pryce J, Hayes B, Goddard M (2012). “Novel strategies to minimize progeny inbreeding while maximizing genetic gain using genomic information.” *Journal of dairy science*, **95**(1), 377–388.
- Pukelsheim F (2006). *Optimal design of experiments*. SIAM.
- Rietveld CA, Medland SE, Derringer J, Yang J, Esko T, Martin NW, Westra HJ, Shakhbazov K, Abdellaoui A, Agrawal A, *et al.* (2013). “GWAS of 126,559 individuals identifies genetic variants associated with educational attainment.” *science*, **340**(6139), 1467–1471.
- Rincent R, Laloë D, Nicolas S, Altmann T, Brunel D, Revilla P, Rodriguez VM, Moreno-Gonzalez J, Melchinger A, Bauer E, *et al.* (2012). “Maximizing the reliability of genomic selection by optimizing the calibration set of reference individuals: comparison of methods in two diverse groups of maize inbreds (*Zea mays* L.).” *Genetics*, **192**(2), 715–728.
- Risch N, Merikangas K, *et al.* (1996). “The future of genetic studies of complex human diseases.” *Science*, **273**(5281), 1516–1517.
- Satman M (2013). “galts: Genetic algorithms and C-steps based LTS (Least Trimmed Squares) estimation.” *R package version*, **1**.
- Schierenbeck S, Pimentel E, Tietze M, Körte J, Reents R, Reinhardt F, Simianer H, König S (2011). “Controlling inbreeding and maximizing genetic gain using semi-definite programming with pedigree-based and genomic relationships.” *Journal of dairy science*, **94**(12), 6143–6152.
- Schölkopf B, Smola AJ (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.
- Schwarz G, *et al.* (1978). “Estimating the dimension of a model.” *The annals of statistics*, **6**(2), 461–464.
- Scrucca L, *et al.* (2013). “GA: a package for genetic algorithms in R.” *Journal of Statistical Software*, **53**(4), 1–37.
- Sivanandam S, Deepa S (2007). *Introduction to genetic algorithms*. Springer Science & Business Media.
- Sun C, VanRaden P, O’Connell J, Weigel K, Gianola D (2013). “Mating programs including genomic relationships and dominance effects.” *Journal of dairy science*, **96**(12), 8014–8023.
- Tendys T (2002). “gafit: Genetic Algorithm for Curve Fitting.” *R package version 0.4*, URL <http://CRAN.R-project.org/src/contrib/Archive/gafit>.

- Tian F, Bradbury PJ, Brown PJ, Hung H, Sun Q, Flint-Garcia S, Rocheford TR, McMullen MD, Holland JB, Buckler ES (2011). “Genome-wide association study of leaf architecture in the maize nested association mapping population.” *Nature genetics*, **43**(2), 159–162.
- Tibshirani R (1996). “Regression shrinkage and selection via the lasso.” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288.
- Turlach BA, Weingessel A (2007). “quadprog: Functions to solve quadratic programming problems.” *R package version*, pp. 1–4.
- VanRaden P, Tassell CV, Wiggans G, Sonstegard T, Schnabel R, Taylor J, Schenkel F (2009). “Invited Review: Reliability of genomic predictions for North American Holstein bulls.” *Journal of Dairy Science*, **92**(1), 16 – 24. ISSN 0022-0302.
- VanRaden PM (2008). “Efficient Methods to Compute Genomic Predictions.” *Journal of Dairy Science*, **91**(11), 4414–23.
- Vapnik V (1998). *Statistical learning theory*. 1 edition. Wiley. ISBN 0471030031.
- Verzani J (2015). *UsingR: Data Sets, Etc. for the Text "Using R for Introductory Statistics"*, Second Edition. R package version 2.0-5, URL <https://CRAN.R-project.org/package=UsingR>.
- Wahba G (1990). *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611970128. <http://epubs.siam.org/doi/pdf/10.1137/1.9781611970128>, URL <http://epubs.siam.org/doi/abs/10.1137/1.9781611970128>.
- Welch WJ (1982). “Branch-and-bound search for experimental designs based on D optimality and other criteria.” *Technometrics*, **24**(1), 41–48.
- Willighagen E (2005). “Genalg: R based genetic algorithm.” *R package version 0.1*, **1**.
- Witten DM, Tibshirani R, Hastie T (2009). “A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis.” *Biostatistics*, p. kxp008.
- Wray N, Goddard M (1994). “MOET breeding schemes for wool sheep 1. Design alternatives.” *Animal Production*, **59**(01), 71–86.
- Yang J, Benyamin B, McEvoy BP, Gordon S, Henders AK, Nyholt DR, Madden PA, Heath AC, Martin NG, Montgomery GW, Goddard ME, Visscher PM (2010). “Common SNPs explain a large proportion of the heritability for human height.” *Nature Genetics*, **42**(7), 565–569. doi:10.1038/ng.608. URL <http://dx.doi.org/10.1038/ng.608>.

Yates F (1935). “Complex experiments.” *Supplement to the Journal of the Royal Statistical Society*, **2**(2), 181–247.

Yu X, Li X, Guo T, Zhu C, Wu Y, Mitchell SE, Roozeboom KL, Wang D, Wang ML, Pederson GA, *et al.* (2016). “Genomic prediction contributing to a promising global strategy to turbocharge gene banks.” *Nature Plants*, **2**, 16150.

Appendices

1. Initial solutions and an island model A.1 Default design criteria implemented in STPGA.

A.2 Using initial solutions and implementing an island model using GA. The scenario in the island model is not necessarily fixed, ingenious or advanced. It is for demonstration purposes and the users are encouraged to play around with it or change it completely.

Box A1: Loading the wheat data set included in STPGA

```
> data(WheatData)
> svdWheat<-svd(Wheat.K, nu=50, nv=50)
> PC50WHeat<-Wheat.K%*%svdWheat$v
> rownames(PC50WHeat)<-rownames(Wheat.K)
> DistWheat<-dist(PC50WHeat)
> TreeWheat<-cutree(hclust(DistWheat), k=4)
> Test<-rownames(PC50WHeat)[TreeWheat==2]
> Candidates<-setdiff(rownames(PC50WHeat), Test)
> deptest<-Wheat.Y[Wheat.Y$id%in%Test,]
> Ztest<-model.matrix(~-1+deptest$id)
```

Box A2: Function that implements a simple island model

```

> repeatgenalg<-function(numrepsouter,numrepsinner){
+   StartingPopulation2=NULL
+   for (i in 1:numrepsouter){
+     StartingPopulation<-lapply(1:numrepsinner, function(x){
+       GenAlgForSubsetSelectionNoTest(P=PC50WHeat,
+         ntoselect=50, InitPop=StartingPopulation2,
+         npop=200, nelite=5, mutprob=.5, mutintensity = rpois(1,1),
+         niterations=200,minitbefstop=5, tabumemsize = 0, tabu=FALSE,
+         plotiters=FALSE, lambda=1e-9,
+         errorstat="CDMEAN", mc.cores=4)})
+     StartingPopulation2<-vector(mode="list", length = numrepsouter*1)
+     ij=1
+     for (i in 1:numrepsinner){
+       for (j in 1:1){
+         StartingPopulation2[[ij]]<-StartingPopulation[[i]][[j]]
+         ij=ij+1
+       }
+     }
+   }
+   ListTrain<-GenAlgForSubsetSelectionNoTest(
+     P=PC50WHeat[rownames(PC50WHeat)%in%Candidates,],
+     ntoselect=50,
+     InitPop=StartingPopulation2, npop=200,
+     nelite=10, mutprob=.5, mutintensity = 1, niterations=200,
+     minitbefstop=50, tabumemsize = 0, tabu=FALSE, plotiters=FALSE,
+     lambda=1e-9, errorstat="DOPT", mc.cores=4)
+   return(ListTrain)
+ }

```

Box A3: Function that implements a simple island model

```

> ListTrain<-repeatgenalg(10, 4)
> min(ListTrain$`Best criterion values over iterarions`)

[1] -47.44094

> min(Train4$`Best criterion values over iterarions`)

[1] 5.123539

> deptrainopt<-Wheat.Y[(Wheat.Y$id%in%ListTrain$`Solution with rank 1`),]
> Ztrain<-model.matrix(~-1+deptrainopt$id)
> modelopt<-emmreml(y=deptrainopt$plant.height,
+   X=matrix(1, nrow=nrow(deptrainopt), ncol=1),
+   Z=Ztrain, K=Wheat.K)
> predictopt<-Ztest%*%modelopt$uhat
> cor(predictopt, deptest$plant.height)

[,1]
[1,] 0.3155026

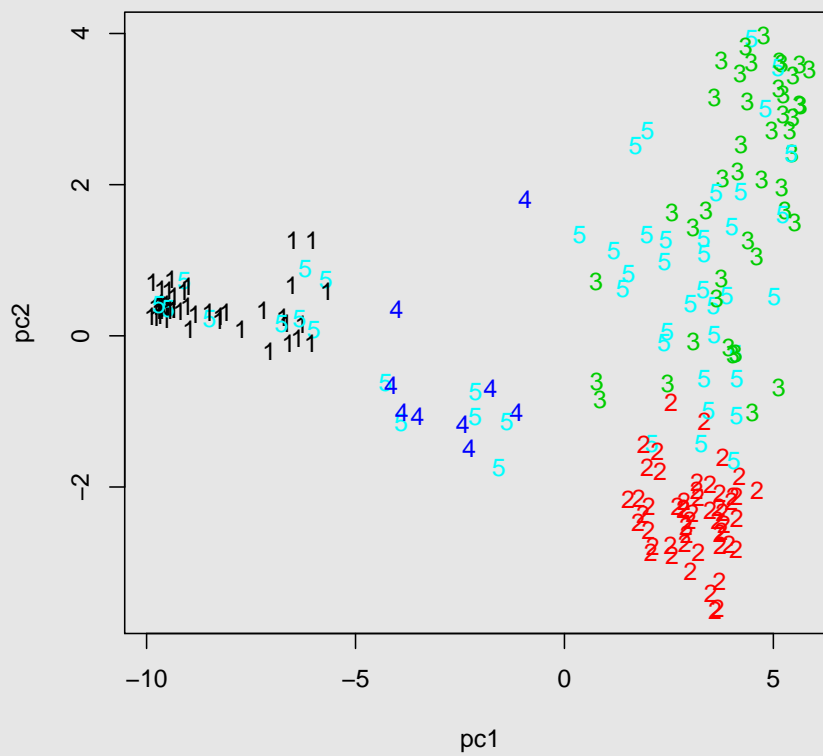
```


Box A4: Function that implements a simple island model

```
> TreeWheatTrain<-TreeWheat
> TreeWheatTrain[names(TreeWheatTrain)%in%ListTrain$`Solution with rank 1`]<-5
```

Box A5: Function that implements a simple island model

```
> plot(PC50Wheat[,1],PC50Wheat[,2], col=TreeWheatTrain,
+      pch=as.character(TreeWheatTrain), xlab="pc1", ylab="pc2")
```

**A.3 A mixed Integer quadratic programming function for proportions**

Box A6: A mixed Integer quadratic programming function for proportions

```

> require(quadprog)
> MIQP<-function(Dmat, dvec, cardinality, npop=200,
+               nelite=10, mutprob=.5, mutintensity = 1, niterations=200,
+               minitbefstop=50, tabumemsize = 1, plotiters=FALSE, tabu=FALSE,
+               lambda=1e-5, mc.cores=4){
+   P<-cbind(dvec,Dmat)
+   rownames(P)<-rownames(Dmat)
+   STPGAUSERDEFFUNC<-function(Train,Test=NULL, P, lambda=1e-5, C=NULL){
+     smallD<-P[rownames(P)%in%Train,-1]
+     smallD=smallD+lambda*diag(nrow(smallD))
+     smallD<-P[rownames(P)%in%Train,1]
+     n=length(smallD)
+     sol <- solve.QP(Dmat=smallD,
+                     dvec=smallD, Amat=cbind(rep(1,n),diag(n),-diag(n)),
+                     bvec=rbind(1,matrix(0,ncol=1,nrow=n),matrix(-1,ncol=1,nrow=n)),
+                     meq=1)
+     names(sol$solution)<-rownames(smallD)
+     return(sol$value)
+   }
+   GAOUT<-GenAlgForSubsetSelectionNoTest(P=P, ntoselect=cardinality, npop=npop,
+   nelite=nelite, mutprob=mutprob,
+   mutintensity = mutintensity, niterations=niterations,
+   minitbefstop=minitbefstop, tabumemsize = tabumemsize,
+   plotiters=plotiters, tabu=tabu,
+   lambda=lambda, errorstat="STPGAUSERDEFFUNC",
+   mc.cores=mc.cores)
+   smallD<-P[rownames(P)%in%GAOUT$`Solution with rank 1`, -1]
+   smallD=smallD+lambda*diag(nrow(smallD))
+   smallD<-P[rownames(P)%in%GAOUT$`Solution with rank 1`, 1]
+   n=length(smallD)
+   sol <- solve.QP(Dmat=smallD, dvec=smallD,
+                   Amat=cbind(rep(1,n),diag(n),-diag(n)),
+                   bvec=rbind(1,matrix(0,ncol=1,nrow=n),
+                                   matrix(-1,ncol=1,nrow=n)), meq=1)
+   names(sol$solution)<-rownames(smallD)
+   return(sol$value)
+ }

```

Affiliation:

Deniz Akdemir

Department of Epidemiology and Biostatistics

Michigan State University

East Lansing MI USA

E-mail: deniz.akdemir.work@gmail.com

Table 1: Default design criteria implemented in STPGA

critterion name	formula	type	equation
<i>AOPT</i>	$trace[C(X'_{Train}X_{Train} + \lambda * I)^{-1}C']$	X	Equation 1
<i>CDMAX</i>	$max[diag(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C') / diag(CX_{Test}X'_{Test}C')]$	X	Equation 2
<i>CDMAX0</i>	$max[diag(CX_{Train}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Train}C') / diag(CX_{Train}X'_{Train}C')]$	X	Equation 2
<i>CDMAX2</i>	$max[diag(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Train}X_{Train}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C') / diag(CX_{Test}X'_{Test}C')]$	X	Equation 2
<i>CDMEAN</i>	$mean[diag(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C') / diag(CX_{Test}X'_{Test}C')]$	X	Equation 2
<i>CDMEAN0</i>	$mean[diag(CX_{Train}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Train}C') / diag(CX_{Train}X'_{Train}C')]$	X	Equation 2
<i>CDMEAN2</i>	$mean[diag(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Train}X_{Train}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C') / diag(CX_{Test}X'_{Test}C')]$	X	Equation 2
<i>CDMEANMM</i>	$-mean[diag(CZ_{Test}(K - lambda * (Z'_{Train}MZ_{Train})^{-1} + \lambda * Kinv)Z'_{Test}C') / (diag(CZ_{Test}KZ'_{Test}C'))], M = I - W(W'W)^{-1}W'$	K	Equation 5
<i>DOPT</i>	$logdet(C(X'_{Train}X_{Train} + \lambda * I)^{-1}C')$	X	Equation 1
<i>EOPT</i>	$max(eigenval(C(X'_{Train}X_{Train} + \lambda * I)^{-1}C'))$	X	Equation 1
<i>GAUSSMEANMM</i>	$-mean(diag(Z_{Test}KZ'_{Test} - Z_{Test}KZ'_{Train}(Z'_{Train}KZ'_{Train} + \lambda * I)^{-1}Z_{Train}KZ'_{Test}))$	K	Equation 3
<i>GOPTPEV</i>	$max(eigenval(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C'))$	X	Equation 2
<i>GOPTPEV2</i>	$mean(eigenval(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C'))$	X	Equation 2
<i>PEVMAX</i>	$max(diag(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C'))$	X	Equation 2
<i>PEVMAX0</i>	$max(diag(CX_{Train}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Train}C'))$	X	Equation 2
<i>PEVMAX2</i>	$max[diag(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Train}X_{Train}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C')]$	X	Equation 2
<i>PEVMEAN</i>	$mean(diag(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C'))$	X	Equation 2
<i>PEVMEAN0</i>	$mean(diag(CX_{Train}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Train}C'))$	X	Equation 2
<i>PEVMEAN2</i>	$mean[diag(CX_{Test}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Train}X_{Train}(X'_{Train}X_{Train} + \lambda * I)^{-1}X'_{Test}C')]$	X	Equation 2
<i>PEVMEANMM</i>	$mean(diag(CZ_{Test}(Z'_{Train}MZ_{Train} + \lambda * Kinv)^{-1}Z'_{Test}C')) / M = I - W(W'W)^{-1}W'$	K	Equation 4